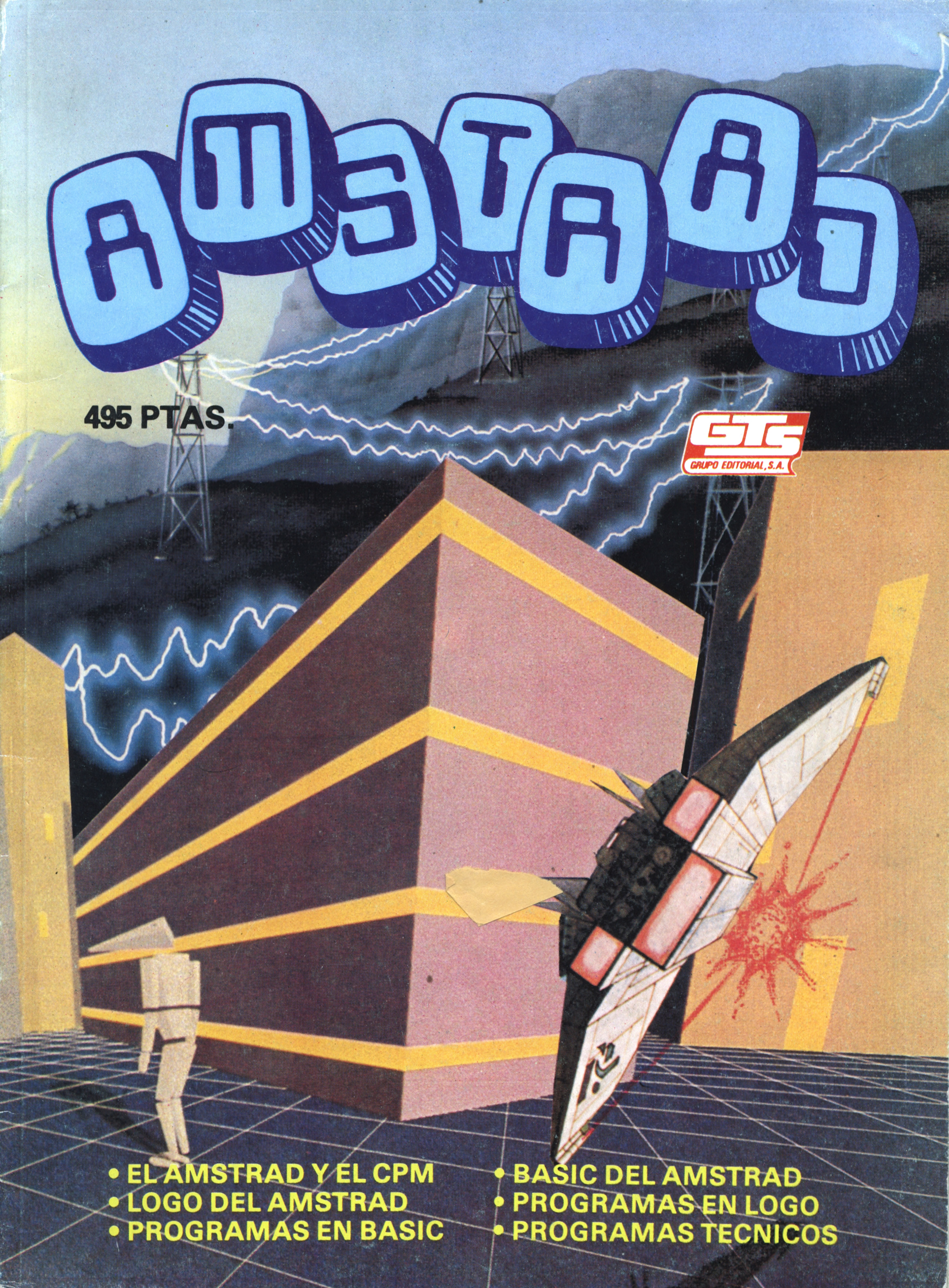

AMSTRAD
495 PTAS.
GTS
GRUPO EDITORIAL, S.A.
• EL AMSTRAD Y EL CPM
• LOGO DEL AMSTRAD
• PROGRAMAS EN BASIC
• BASIC DEL AMSTRAD
• PROGRAMAS EN LOGO
• PROGRAMAS TECNICOS

# EDITORIAL

*En los últimos meses ha alcanzado el AMSTRAD cotas de estimable difusión, hecho que constituye por sí mismo motivo más que sificiente para que le dedicásemos toda nuestra atención. Por ello, salimos con AMSTRAD EDUCATIVO.*

*En el quiosco, veremos, una revista más para el usuario, una nueva manera y un enfoque de como utilizar el AMSTRAD*

*Con una simple mirada al sumario, en esta misma página, podéis haceros una idea de la trayectoria que pretendemos seguir. No obstante, como la revista está pensada y hecha para vosotros, sería muy importante que participáseis en ella dando vuestra opinión, haciendo todo tipo de sugerencias, indicando, en una palabra, como queréis que sea vuestra revista.*

*Esperamos vuestras opiniones. Que os guste.*

# SUMARIO







**Edita:** Grupo Editorial G.T.S., S. A.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02.
**Director:** Antonio Bellido.
**Colaboran** en este número:
Juan M. Pintor, Víctor J. Campo López.
**Maquetación:** Susana M. Villalba.
**Secretaria de Redacción:** Mercedes Jamart.
**Publicidad:** Dpto. propio.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02

**Fotocomposición:**
Fotocom, S. A.
**Fotomecánica:**
La Unión
**Imprime:**
Gráficas FUTURA Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23
Políg. Ind. Cobo Calleja
FUENLABRADA (Madrid)
**Distribuye:**
R.B.A. Promotora de Ediciones, S. A.
Travesera de Gracia, 56 Atico, 1.ª.
Tfno. 93 - 200 82 56
**Depósito Legal:**
M. 8.904-1986

# EL AMSTRAD Y EL CP/M

## *Función del DOS en un microprocesador. Puesta en marcha*

DANDO por supuesto que el lector ha utilizado un ordenador, tal vez se haya preguntado cómo es posible que al pulsar una tecla, aparezca un carácter en la pantalla.

Bien, esto es así debido a que en algún lugar de la memoria, hay un programa cuya función es examinar constantemente el teclado, informando a la CPU de todas las alteraciones que se produzcan, y, si las hay, determinar qué tecla fue apretada, fijar el modelo de puntos a que corresponde el carácter equivalente, y activar el tubo de rayos catódicos para su impresión en pantalla.

Y, admitiendo que el comando SAVE del BASIC encargado de transferir la información de la RAM a un soporte magnético se haya empleado en alguna ocasión, cabría plantearse: ¿Cómo se produce la transferencia? y ¿qué lo controla y adapta? Bajo las preguntas anteriores, y los procesos implicados en las respuestas, está subyacente el sistema operativo.

El sistema operativo es un conjunto de programas que supervisa constantemente todo el equipo, y sin el cual la máquina es inservible e incapaz de efectuar maniobra alguna.

Algunos sistemas operativos están grabados en ROM, especialmente en los microordenadores de menor porte, encargándose de los controles necesarios. Esto quiere decir que desde el momento de la conexión, el ordenador está bajo el control de su sistema operativo. En el otro extremo están aquellos computadores que deben transferir de un disco a la memoria interna su sistema operativo (DOS). Es claro pues, que en este último caso y en el momento de la conexión, el computador está imposibilitado para actuar, incluso en el sentido de manejar el disco que contiene el DOS que necesita.

Para superar esta inconsecuencia, un pequeño programa grabado en ROM es el encargado de localizar el sistema operativo en el disco situado en una unidad de disco predeterminada, e iniciar la carga del DOS, cediendo el control a este.

Este proceso es conocido como **booting** o boot. Nosotros nos referimos a él mediante la expresión "inicializar el sistema" o, también, como una entrada o arranque en frío **(cold start).**

Un DOS ofrece más opciones al usuario de un ordenador, que los sistemas operativos en ROM, teniendo a su cargo, además de la supervisión general, una serie de funciones para manipular los ficheros almacenados en un disco y la unidad donde éstos están instalados.

Estas funciones se cumplen gracias a ciertos comandos u órdenes que permiten al operador instruir al DOS en el sentido adecuado.

El estudio de estos comandos se verá en la segunda parte de este curso.

Como ejemplo de las misiones encomendadas a estos comandos se pueden anticipar la orden de mostrar en pantalla un directorio completo de los ficheros que contiene un disco y la detención de un proceso ya iniciado.

Un sistema operativo elemental apenas estará capacitado, por ejemplo, para cargar en RAM un fichero determinado, buscándolo entre todos los almacenados en el soporte en que esté contenido. Por el contrario, un DOS del tipo de CP/M puede averiguar exactamente donde está situado cualquier fichero en el soporte.



POR A. BELLIDO

Por todo lo expuesto, y sea cual sea su grado de sofisticación, un sistema operativo puede considerarse como un conjunto de programas que, controlando la información dentro del ordenador, permiten la comunicación entre éste y el operador.

CP/M se hace cargo, además, de la actuación de los diversos periféricos.

## *Compatibilidad*

Con el término inglés **portability** se da a entender la capacidad de un programa para ser ejecutado o explotado por más de un tipo de ordenador.

Esta idea la expresamos en español con la palabra compatibilidad.

Dicho esto, pasemos a considerar los motivos por los cuales un DOS es, o no, compatible en dos ordenadores diferentes, y la razón, por la cual distintos DOS no pueden actuar sobre el mismo ordenador.

Con anterioridad, en varias ocasiones, se ha hecho referencia al DOS como a un conjunto de programas.

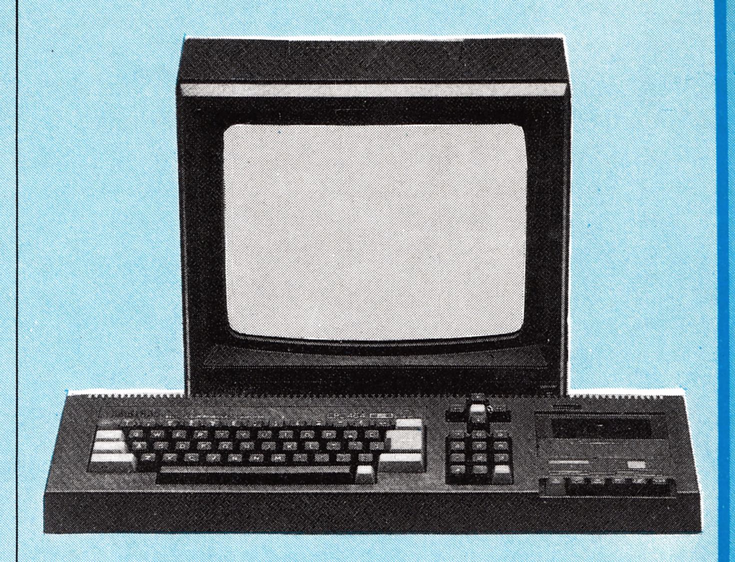
Estos programas deben estar escritos en un determinado lenguaje de programación.

El lenguaje de programación, como se comenta en otro lugar, está en función del microprocesador encargado de ejecutar sus instrucciones.

Por tanto, un DOS desarrollado para correr sobre procesadores del tipo 8080 A, Z 80, y 8085, como CP/M-80, no está en condiciones de servir en un equipo que porte un 8086 o un 8088, para los cuales sería necesario emplear CP/M-86 o MS-DOS. Y, aún entre estos últimos, su distinta concepción los hace diferentes. Antes de proseguir, tomemos contacto con las tres tareas fundamentales que tienen encomendadas los sistemas operativos en disco de los tipos CP/M y MS-DOS:

* **Recibir y ejecutar los comandos u órdenes,** a los que ya se ha dedicado algún párrafo y sobre los que volveremos detalladamente.

* **Controlar la organización de los datos** contenidos en un disco. Esta actividad es invariable para un mismo DOS.



* **Control de entradas y salidas.** Es una parte del DOS que contiene todas las rutinas necesarias para comunicar el microprocesador con los periféricos. Cada fabricante de ordenadores lo configura de acuerdo con su propio criterio, y por tanto puede variar en un mismo tipo de DOS, según haya sido, o no, modificada esta sección del sistema operativo.

La causa de incompatibilidad entre sistemas operativos en disco del mismo tipo depende por tanto, de las alteraciones introducidas en las rutinas de control de entrada/salida por los fabricantes para adaptarlas a sus respectivas configuraciones de ordenador.

Por consiguiente las causas de incompatibilidad entre un ordenador y un programa desarrollado en base a un determinado DOS pueden tener los siguientes orígenes:

1.° **El DOS bajo el cual fue desarrollado el programa es distinto del que controla las operaciones de la máquina.** Esta causa es evidente ya que el conjunto de rutinas que conforman un DOS de una determinada marca, son distintas de las de otro cualquiera. Tal sería el caso de un programa escrito para CP/M-86, el cual sería incompatible en un ordenador con MS-DOS.

2.° **El DOS bajo el cual fue desarrollado el programa, es del mismo tipo del que controla las operaciones de la máquina,** pero el programa no es compatible por que el control de entradas y salidas no es el mismo en ambos ordenadores.

3.° **El DOS bajo el cual fue desarrollado el programa "corre" es un microprocesador distinto del que porta el ordenador.** Este es el caso de aquellos programas escritos, por ejemplo, para un CP/M-80, que trabaja correctamente en Z-80, 8080 y 8085, siendo incompatible, con otro tipo de microprocesador, como sería el caso del 8086 y 8088, que requieren un DOS del tipo CP/M-86 o MS-DOS.

## *Origen de CP/M. Su desarrollo. Versiones*

El origen de CP/M está situado en los intrincados comienzos de los modernos ordenadores personales, los cuales, por cierto, se gestaron sin que nadie tuviera intención de darlos a la luz.

La historia, resumida, es esta.

Hacia 1971, las compañía californianas dedicadas a manufacturar circuitos integrados estaban persiguiendo la idea de llegar a producir microprocesadores económicos cuya misión fuera controlar los mecanismos de una fábrica o automatizar ciertos procesos, como por ejemplo, los arranques y paradas de un ascensor.

En otras palabras, buscaban un capataz electrónico para dirigir el trabajo de otras máquinas.

Este esfuerzo tuvo como consecuencia la aparición del primer microprocesador, denominado 4004 por su creador —INTEL—. De 4 *bits*, con muy pequeña capacidad operativa.

En 1972, INTEL desarrolló el 8008 de 8 *bits*, gracias al cual se abrieron las puertas a los entendidos para montar sus propios computadores, sin posibilidad de conectarlos con teclados o pantallas.

INTEL continuó su saga con el microprocesador 8080 también de 8 *bits*, pero manejando hasta 64 *Kbytes* de memoria interna.

Por estas fechas, Gary Kildall era consultor de INTEL trabajando para **Microprocessor Application Associates** en un proyecto de lenguaje de programación —nombrado PL/M— para el más reciente microprocesador de INTEL. En esta situación, Kildall pensó que era posible unir un microprocesador, de las características del que tenía entre manos, con una unidad de disco de Shugart y a un teletipo. El fin que perseguía era dotar a cada ingeniero de INTEL de su propio ordenador individual, en lugar de compartir entre ellos un ordenador de mayores prestaciones. La propueta fue rechazada.

Como respuesta, Kildall y un amigo —Torode—, construyeron su primer sistema. Aquel se encargó del sistema lógico, este del físico.

De aquí salió un programa monitor, o guía, de las operaciones encargadas a la CPU para controlar los periféricos que tenía a su cargo. Su nombre: CP/M, acrónimo de **Control Program/Monitor.**

Digital Research se creó para su comercialización.

Este fue el primer sistema operativo para el nuevo concepto de "microordenador", que, fue aceptado por aquellos fabricantes que querían completar sus equipos con unidades de disco, y cuyo computador estaba basado en un microprocesador 8080.

Posteriormente, y con mayor rapidez de proceso, CP/M correría igualmente sobre el 8085 de INTEL y el Z80 de Zilog.

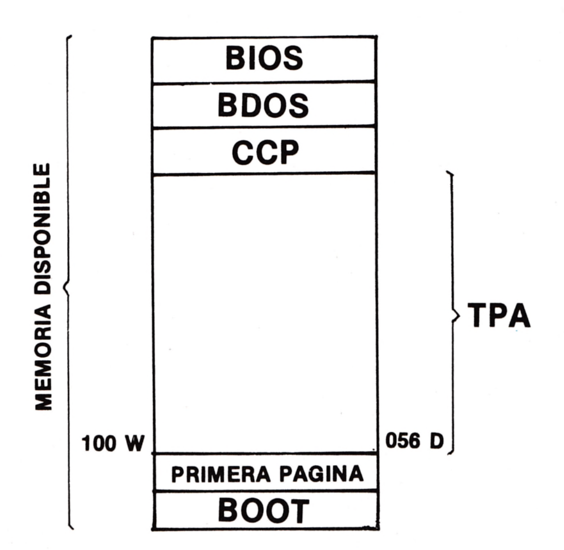
En 1981, Digital Research puso en el mercado CP/M 86, un DOS para ordenadores basados en microprocesadores 8086 y 8088.

Las versiones de CP/M-80 lanzadas al mercado hasta el momento son 1.3, 1.4, 2.0, 2.1 y 2.2.

La norma general para interpretar los dígitos que dan nombre a cada versión, es considerar que el número situado a la izquierda del punto se refiere a la propia versión y el primer dígito a la derecha del punto indica las diferentes revisiones de una versión.

Un segundo dígito a la derecha del punto, indica una adaptación específica a un ordenador de la versión y revisión manifestada según lo explicado anteriormente.

En la actualidad, el número de usuarios de CP/M debe pasar del millón, y la mayoría de los ordenadores utilizan este DOS, bien como elemento base del equipo o bien como opción.




## *Disco del sistema (System Disk)*

Con anterioridad, nos hemos referido a un DOS como a un conjunto de programas, y CP/M no es una excepción.

El disco que contiene los programas de un DOS, recibe la denominación genérica de "disco del sistema" (*System disk*).

En un disco del sistema correspondiente a CP/M las dos pistas más externas, al menos, están reservadas para los programas que lo integran, los cuales se agrupan bajo las siguientes denominaciones, y con las funciones que se indican:

* **BIOS** *(Basic input/output system)* o "sistema básico de entradas y salidas" entre la CPU y el resto de los periféricos. Las misiones generales que tiene asignadas son:

1.° Localizar el periférico emisor o receptor de información, y comprobar que está disponible.

2.° Efectuar la transmisión de la información con eficacia.

3.° Detectar errores en la transmisión y fallos en el sistema físico.

* **BDOS** *(Basic disk operating system)* o "Sistema básico operativo del disco", destinado a controlar los ficheros —y sus datos— contenicos en un disco. En este sentido, al añadir un nuevo fichero a un determinado disco, el BDOS se encarga de:

1.° Detectar que en la pista catálogo no existe otro fichero con el mismo nombre.

2.° Detectar que hay espacio suficiente en el disco, para contener el nuevo fichero.

3.° Dar entrada en la pista catálogo al nuevo fichero y sus parámetros.

* **CCP** *(Console command processor)* o "procesador de órdenes de consola", gracias al cual, el usuario puede comunicarse con el DOS a través del teclado, escribiendo en él las diferentes órdenes que pueden ser dadas, y que se estudian posteriormente. Cuando CP/M está en condiciones de aceptar estas órdenes, decimos que el sistema está a "nivel de operador".

* **BOOT** es un pequeño programa destinado a cargar la totalidad de CP/M en la memoria interna, transfiriendo el control al BIOS, el cual lo transfiere, finalmente, al CCP.

## *Primeros pasos con CP/M*

Una vez instalado el ordenador de acuerdo con las condiciones dadas por el fabricante, el usuario interesado en el DOS debe recurrir al manual, o sección del manual, dedicado al CP/M. Lo usual será encontrarse con una GUIA DEL USUARIO O UNA INTRODUCCION A LAS CARACTERISTICAS Y FACILIDADES DEL CP/M.

Estos manuales, además de una somera descripción del CP/M y su función, comienzan dando unas normas para la correcta lectura e interpretación del contenido de los mismos, y el proceso que debe seguirse para transferir CP/M del disco a la RAM.

La carga del CP/M puede diferir bastante de un equipo a otro, por lo cual se hace imprescindible seguir las instrucciones dadas, a este respecto, en el manual. No obstante, es usual uno de estos procedimientos:

1. Conectar todos los dispositivos y, a continuación, colocar el disco conteniendo CP/M en una determinada unidad de disco. Al cerrar la solapa de seguridad de dicha unidad se produce la carga automática del DOS.

2. Instalar el disco conteniendo CP/M en la unidad de disco oportuna y posteriormente conectar la máquina, en cuyo momento se inicia la transferencia del DOS a la RAM.

Otros computadores, como el AMSTRAD, con algún tipo de sistema operativo en ROM, exigen escribir, por ejemplo, CPM seguido del correspondiente RETURN o ENTER para transferir el DOS a la RAM, o pulsar un determinado grupo de teclas simultáneamente.

La unidad de disco —en el supuesto de haber varios disponibles— donde se debe instalar CP/M para proceder a la carga anteriormente apuntada se denomina unidad por defecto, o unidad por omisión *(default drive)* inicial y su nombre: unidad A.

Los nombres que reciben las diferentes unidades de disco que un computador tiene conectadas constantemente, o en línea *(on line)*, suelen ser A, B, C, D y así sucesivamente, en orden alfabético, una letra por cada unidad disponible.

La versión 2.0 y posteriores de CP/M pueden soportar hasta 16 unidades de disco.

El proceso de carga de CP/M se da por concluido cuando en la pantalla aparece el nombre de la unidad de disco encargada de la transferencia seguido del símbolo >. Así:

A >

Esta impronta *(pronpt)* indica al usuario que el DOS está a nivel de operador *(system-level)*. En otras palabras, CP/M espera recibir algún tipo de orden a través del teclado.

Una vez hayamos colocado el sistema a este nivel, como medida preventiva y sin mayores conocimientos, debemos sacar una copia del disco que contiene el original CP/M y que acompaña al equipo, siguiendo las instrucciones que al efecto, sin duda, están redactadas en el manual de operaciones del ordenador.

El original debe ser puesto a buen recaudo, y utilizar la copia así obtenida para los trabajos alrededor de CP/M.

Dejamos el tema en estas condiciones y pasemos al siguiente epígrafe.

## *Ubicación de CP/M en la memoria interna*

En este punto, vamos a admitir que CP/M ha sido transferido del disco a la memoria interna del computador, con lo cual, y aparte de otros mensajes que varían de máquina a máquina, tendremos en pantalla el *prompt* indicativo de que el DOS está a nivel de operador (A>).

Ahora, CP/M está debidamente encajado en la RAM, de acuerdo con la siguiente estructura:

Los conceptos de BIOS, CCP y BOOT ya nos son conocidos del epígrafe anterior, y los programas que comprenden se sitúan en la RAM en las posiciones de memoria relativas que indica el esquema.

En la misma figura, se observa que la primera página de memoria, se la reserva CP/M para su uso, y que dentro de ella está situado BOOT. El significado de "página de memoria" se estudiará en su momento. (Ver SAVE).

La zona de memoria denominada **TPA** *(Transient program area)* o "área de programas transitorios" está comprendida entre la primera página y el comienzo de CCP. En esta zona se ubican los programas que han de ser ejecutados por el ordenador, bajo el control de CP/M.

El TPA dispone de tanta memoria como resulte de restar de la cantidad de memoria disponible, la requerida por el BIOS más el BDOS y el CCP, según se aprecia en el esquema. En otras palabras, el TPA es la memoria disponible tras cargar CP/M en el computador.

El bloque de memoria ocupado por BIOS, BDOS y CCP puede ser desplazado, dentro de lo que permita la RAM disponible, para ajustarse a las exigencias del usuario.

Como es lógico, una vez finalizada la carga, el DOS queda a la espera de recibir instrucciones sobre la acción a llevar a efecto o, dicho de otro modo, el sistema está bajo el control del CCP.

A esta situación nos referiremos en ocasiones diciendo que estamos a nivel operador, como ha sucedido en un par de ocasiones.

En este momento, ya tenemos una aproximación a lo que implica estar a nivel operador.

# Los ficheros en el AMSTRAD

POR A. BELLIDO

## Cómo crear ficheros en disco

***A** lo largo de las páginas que siguen, doy por supuesto un conocimiento no profundo pero sí elemental de la programación en BASIC.*

***El objetivo encomendado a este libro, es ayudar al lector interesado en los temas que aquí se tratan a dominar el dispositivo periférico que más eficacia va a añadir a su computador: la unidad de disco.***

AUN cuando **Cómo crear ficheros en disco** está dedicado a la creación y mantenimiento de ficheros en disco, la primera parte trata sobre los conceptos previos generales que todo usuario de ordenador debe conocer para estar en condiciones de comprender los mecanismos que permiten a la máquina funcionar de acuerdo con los deseos del programador.

En este sentido, el principal problema con que hemos de toparnos, es concebir en su verdadera dimensión cada una de las palabras, conceptos y órdenes que el manejo de ficheros en disco hace necesario emplear.

Las personas que sepan inglés tendrán una ventaja añadida que yo no tengo en cuenta, ya que considero fundamental transmitir a nuestro idioma toda la estructura mental que soporta a cada palabra o expresión anglosajona que nos veamos obligados a utilizar.

En la segunda parte del curso se tratan específicamente los ficheros en BASIC tanto secuenciales como directos.

Todos los epígrafes contienen ejemplos que contribuyen a afianzar las ideas que en ellos se explican, y se cierran con ejercicios que permiten al lector evaluar sus conocimientos hasta el momento.

Los ejercicios de por sí tienen un gran interés didáctico en función de que ciertos detalles, no dichos explícitamente en el texto, tienen la oportunidad de aparecer reflejados en ellos.

Por otra parte, cuando sea conveniente recurrir a acrónimos, se utilizarán los derivados de las expresiones originales, por ser estos los que suelen aparecer en las publicaciones en español y, desde luego, en las anglosajonas. En todo caso, los considero como unos sustantivos nuevos aplicados a unas ideas nuevas, ya que, de utilizar otro criterio, sólo redundaría en aumentar el léxico, ya confuso, actualmente existente.

Por fin, y con respecto a la exposición de los temas, he tratado de utilizar un lenguaje sencillo, con sentencias breves y claras, apoyado, cuando ha lugar, en esquemas y gráficos que permitan una mejor comprensión.

### *DISK OPERATING SYSTEM*

Esta expresión anglosajona y su acrónimo DOS son términos usuales en la literatura que trata los asuntos que aquí nos ocupan.

Para situarnos en el contexto, y dejando atras precisiones de momento, podemos decir

que un DOS es un conjunto de programas que tiene por misión controlar el tráfico de información en el ordenador.

Desde el punto de vista semántico, un DOS puede entenderse, bien como un sistema operativo **de** disco o bien como un sistema operativo **en** disco.

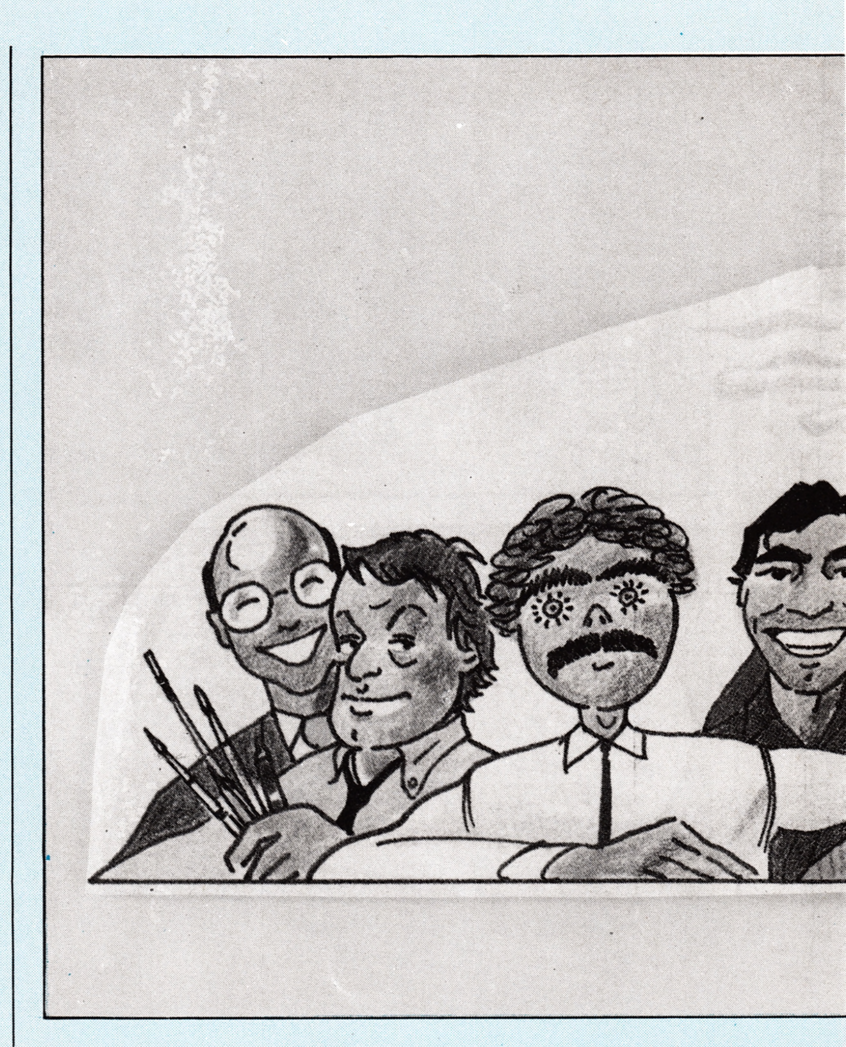
La primera acepción es válida dado que, en sus orígenes, el DOS estaba encargado fundamentalmente, de manejar el disco. Hoy en día, como veremos, el DOS tiene asignadas muchas más tareas.

El segundo significado es igualmente válido, ya que los programas que componen el DOS están contenidos en disco.

Nosotros aquí, al referirnos a este concepto, utilizaremos el término DOS por ser el más generalizado en el terreno informático.

Dicho esto, es conveniente precisar el sentido que debemos darle a palabras tales como información y ordenador, aparecidas hasta el momento, y otras que surgirán como consecuencia de sus definiciones.

A este menester, de extraordinaria importancia para la consecución de los objetivos últimos que nos hemos propuesto, se dedicarán los próximos epígrafes.



## *INFORMACION. CONCEPTO*

En líneas generales, información es una comunicación de conocimientos o, en términos informáticos, transmisión de datos.

La unidad elemental de información es el **bit.**

La palabra **bit** es una contracción de *binary digit* y alude a cualquiera de los dos dígitos que componen el sistema de numeración en base dos: 0 y 1.

Otra unidad de información, de orden superior, es el **byte.**

Un *byte* es un grupo de ocho bits o, lo que es igual, cualquier combinación de ocho ceros y unos, del tipo 10011101.

Con un *bit* tenemos dos posibilidades de recibir o emitir información: 0 ó 1; y si admitimos un código que asocie el **cero** a NO (o FALSO) y el **uno** a SI (o VERDADERO), habremos establecido una forma elemental de comunicación.

Así, si preguntamos a algo o a alguien: ¿Aprobó Juan Pérez?", y esto o éste nos responde: 1, sabremos que Juan Pérez SI aprobó.

Es claro, que con un *bit* sólo podemos informar de un acontecimiento entre dos posibles.

Pero, ¿qué sucede cuando la información debe abarcar a cualquier evento?

Para los seres humanos, a estas alturas de su evolución, es relativamente sencillo comunicarse a través del lenguaje hablado y, utilizando el código que representa el conjunto de los símbolos que forman el alfabeto, mediante la escritura.

Pero, ¿qué sucede cuando la información debe establecerse entre hombre y máquina o entre dos máquinas?



La. respuesta a esta interesante cuestión pasa por saber que, por exigencias de su electrónica, un ordenador maneja exclusivamente números en base dos.

Cualquier número en base dos tiene su correspondiente en otra base y, más concretamente, en base 10.

El sistema decimal, —o en base 10— es el utilizado normalmente en la vida diaria y, por tanto, a él nos referiremos aun cuando los ordenadores sólo trabajen con números binarios.

Por otra parte, y como ya dijimos, un **byte** es una combinación de ocho **bits** y consiguientemente, representa un número en base dos compuesto por ocho dígitos binarios.

Así, por ejemplo, el binario 00000000 equivale al decimal 0, o el binario 01111110 al decimal 126.

El número más alto que se puede representar mediante un *byte* es 11111111 o, en decimal, 255.

Teniendo esto por cierto, sólo resta preparar un código que dé una correspondencia entre todos los caracteres que son necesarios para la escritura y otros tantos números.

La **American Standard Code for Information Interchange** ha establecido un código que lleva por nombre su acrónimo ASCII, y del cual hay amplia referencia en el manual del AMSTRAD.

Como se puede observar al estudiarlo, los primeros 31 números cumplen las funciones que indican los comentarios, y sirven para completar todos los requisitos que pueden ser exigidos en una comunicación escrita.

A los efecto que aquí se persiguen, la correspondencia entre números y caracteres de uso en la escritura queda establecida entre el 32 y el 127.

Dicho esto, codificar una cadena de caracteres es sencillo. Hagamos la prueba con la palabra COMPUTADOR:

| LETRA | DECIMAL | BINARIO |
|---|---|---|
| C | 67 | 01000011 |
| O | 79 | 01001111 |
| M | 77 | 01001101 |
| P | 80 | 01010000 |
| U | 85 | 01010101 |
| T | 84 | 01010100 |
| A | 65 | 01000001 |
| D | 68 | 01000100 |
| O | 79 | 01001111 |
| R | 82 | 01010010 |

La tercera columna representa la secuencia de *bytes* que el computador transmitiría para enviar el mensaje que, una vez decodificado, diera por resultado la palabra COMPUTADOR.

Concluyendo. Dentro del ordenador sólo se manejan *bytes*, o lo que es igual, secuencias de ocho bits.

Y es mediante *bytes* como se comunica el computador con sus diferentes periféricos.

# EL BASIC DEL AMSTRAD

## Introducción al curso

POR A. BELLIDO

LA palabra BASIC es un acrónimo formado por las iniciales Beginners All-purpose Symbolic Instruccion Code.

El BASIC es un lenguaje de programación de alto nivel que fue desarrollado en el Dartmouth College en los Estados Unidos de América del Norte, y de esa idea inicial surgieron diferentes dialectos, o distintos BASIC, adaptados por cada fabricante a sus máquinas.

El computador AMSTRAD, incorpora un tipo de BASIC muy amplio, completo y generalizado.

Este curso, cuyo INDICE está en el epígrafe siguiente, está dirigido a mostrar al lector las posibilidades que ofrece el BASIC del AMSTRAD, comenzando con los más elementales conceptos y llegando hasta la creación y mantenimiento de ficheros en disco.

Para conseguir este objetivo, cada capítulo tiene una parte teórica con ejemplos relativos a los temas tratados, una ficha-recuadro donde se enfatizan ciertos puntos o se estudian aspectos marginales de interés específico y, finalmente, se proponen ejercicios, cuya solución, a algunos de ellos. Se ofrece en el siguiente número.

Concluiré esta breve introducción con un comentario respecto a los lenguajes de programación en general y a la postura que el programador debe adoptar frente a ellos.

Todo lenguaje requiere de unas normas de escritura (sintaxis) y de un vocabulario (palabras). Una persona que desee expresarse en ese idioma, ha de conocer esa sintaxis y estar al corriente de ese vocabulario.

Con respecto a la organización del curso, anticipo que los temas a estudiar están agrupados en grandes unidades.

En el primero, CONCEPTOS PREVIOS, se informa al lector sobre el significado y aplicación de las palabras e ideas que utilizará en lo sucesivo.

En el segundo, INSTRUCCIONES FUNDAMENTALES, se desarrolla un estudio de programación en BASIC, con el apoyo que dan la propia instrucción que se está analizando, y algunas de las anteriores.

En el tercero, FUNCIONES INCORPORADAS, se amplía el vocabulario BASIC mediante fichas de todas las funciones incorporadas al AMSTRAD. Algunas de las instrucciones que figuran en este apartado no son propiamente funciones, pero figuran en él por razones didácticas y así se hace constar en su lugar; en el INDICE figuran en negritas.

En el cuarto, INSTRUCCIONES RELACIONADAS CON EL TIEMPO, se analizan las posibilidades de control temporal que tiene a su alcance el programador.

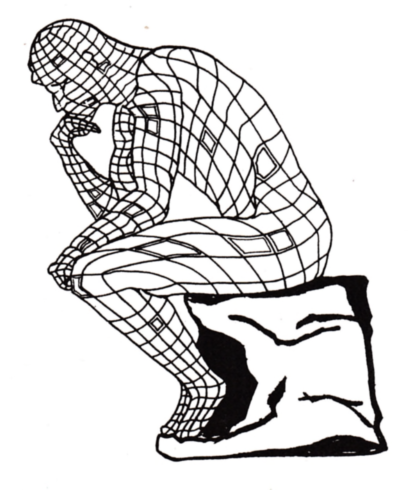
En el quinto, INSTRUCCIONES RELACIONADAS CON LA PRESENTACION, se ve en detalle la tremenda oferta del AMSTRAD para mostrar en pantalla gráficos y mensajes. También la salida a impresora.

En el sexto, MATRICES, se profundiza en el manejo de estos cómodos artificios, conocidos, también, por *arrays*.

En el séptimo, SONIDO, los amantes de la música tienen la oportunidad de conocer, en la práctica, la capacidad del AMSTRAD.

En el octavo, FICHEROS, se introduce al lector en el manejo de discos para el proceso de datos, con una introducción a todos los conceptos que exige la comprensión de la técnica implicada.

## *INDICE DEL CURSO*



# Fichas del

***Uno de los principales problemas con que se enfrenta todo nuevo usuario del AMSTRAD, se deriva de la voluminosidad de su manual.***

***Como una ayuda, y con el fin de sustituir al manual en determinado tipo de consultas, se ofrece a continuación una guía práctica y resumida de los comandos y funciones, que el AMSTRAD pone a disposición del programador en BASIC.***

## *Guía de palabras BASIC*

La notación aplicar, responde a los siguientes criterios:

1. Las palabras encuadradas corresponden a las del BASIC.

2. La información contenida entre dos signos de interjeción (¡...!) **debe ser** suministrada por el programador imprescindiblemente.

3. La información contenida entre dos barras verticales (|...|) **puede ser** suministrada por el programador opcionalmente.

4. Si la palabra **expresión** va entrecomillas, se refiere a una expresión alfanumérica. En caso contrario es una expresión numérica.

AMSTRAD: GUIA DE PALABRAS BASIC. SU SINTAXIS Y APLICACION

*ABS* *¡(expresion)!*

Convierte la **expresión**, sea cual sea su signo, en la misma pero positiva.

PRINT ABS (-1,2)
1,2

*AFTER* *¡expresion 1! |, espresión 2 | gosub*

Llama a una subrutina cuando el tiempo que indica **expresión 1** ha transcurrido. Con

POR A. BELLIDO

**expresión 2** se determina, opcionalmente, el número del retardador: 0, 1, 2, 3. Por defecto es el 0. Cada retardador puede dirigirse a su propia subrutina.

```
5x = 2 : y = 5
10 AFTER x*y*10 GOSUB 30 : AFTER
  x*100, x GOSUB 40
20 FOR N = 0 TO 800 : PRINT "P"; : NEXT
30 PRINT "RO RO RO RO" : RETURN
40 PRINT "R2 R2 R2 R2" : RETURN
```

Después de dos segundos, se interrumpe la impresión del carácter P para dar lugar a las cuatro RO y, tras 4 segundos, a las cuatro R2.

*ASC* *¡("expresion")!*

Se obtiene el código numérico del primer carácter de "expresión

```
PRINT (Ab)
65
```

*ATN* *¡(expresion)!*

Determina el arcotangente de la expresión

```
PRINT ATN (-.41614684)
-0,394348109
```

*AUTO* *| número 1 | |, número 2|*

Hace aparecer en pantalla automáticamente las líneas del listador a partir del **número 1** y con la cadencia dada por **número 2**. Para pasar a la siguiente, basta pulsar ENTER.

Por defecto, **número 1** y **número 2** son 10.

Para cancelar AUTO, apretar ESC.

*BIN$* *¡(EXPRESION 1) |, EXPRESION 2|)!*

Convierte la **expresión 1** en una cadena representativa del binario equivalente, con tantos dígitos como indique **expresión 2.**

Si **expresión 2** no se da, o es demasiado pequeña, la cadena cubre los dígitos binarios mínimos necesarios.

```
10 × = 8 : y = 2040
20 PRINT BIN$ (y/x,x)
11111111
```

*BORDER* *¡expresion 1! |, expresion 2|*

Fia el color dado por **expresión 1** en la zona de pantalla que bordea el área de impresión. Si **expresión 2** se da, el borde cambia de color entre los colores determinados por ambas expresiones, las cuales deben resultar en números comprendidos.
meros comprendidos entre 0 y 26.

```
10 FOR x=0 TO 26
20 BORDER x,26-x
30 FOR y=0 TO 1000 : NEXT
40 NEXT
50 BORDER 1
```

El borde parpadeará —excepto para x=13— entre dos colores.

Pruebe el efecto quitando el segundo parámetro en la línea 20.

*CAT*

Muestra el catálogo de ficheros del disco instalado en el **drive** cuya identificación aparece en pantalla, junto con la capacidad disponible aún. Este comando no afecta al contenido que, a la sazón, tenga el computador.

*CHAIN* *¡"expresion"!|, expresion|*

Carga en memoria, procedente del disco, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**

Si **expresión** no se da, el programa se autoejecuta desde la primera línea del listado.

Carga en memoria, procedente del disco, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**
**? Si expresión** no se da, el programa s Si **expresión** no se da, el programa denominado "expresión" ejecutándose desde la línea dada por **expresión.**
Si **expresión** no se da, el programa se autoejecuta desde la primera línea del listado.

```
x=25 : A$="Laser" : CHAIN A$, x*2
```

Esta secuencia de instrucciones, cargará el programa titulado **Laser** y se autoejecutará a partir de la línea 50.

BASIC

### CHAIN MERGE |"expresion"|, expresion | |, delete número 1 - número 2 |

Actúa exactamente igual que CHAIN, pero mezclando el programa titulado "expresión" con el que, a la sazón, esté en memoria, sustituyendo en éste aquellas líneas de programa que, procedentes de aquel, coincidan.

Si la opción DELETE se usa, todas las líneas comprendidas entre **número 1** y **número 2** quedarán borradas.

CHAIN MERGE A$, 50, DELETE 9000 - 9500

Esta instrucción, cargará en memoria el programa cuyo título está representado por A$, y lo ejecutará a partir de la línea 50, borrando las líneas comprendidas entre 9000 y 9500.

Esta instrucción, cargará en memoria el programa cuyo título está representado por A$, y lo ejecutará a partir de la línea 50, borrando las líneas comprendidas entre 900 y 9500.

### CHR$ ¡(expresion)!

Convierte el número resultante de **expresión**, en su código ASCII equivalente, el cual debe estar comprendido entre 32 (código del espacio) y 255 (código de la doble flecha).

```
x=2 : y=34 : PRINT CHR$ (x*y)
```

### CINT ¡(expresion)!

Convierte la **expresión** en el número entero más próximo

```
PRINT CINT (-1,9), CINT (-1,5), CINT (-1,1)
da:   -2     -2     -1
```

### CLEAR

Borra todas las variables, matrices y funciones definidas por el usuario. Cancela cualquier situación de OPEN en un fichero.

### CLEAR INPUT

Asegura que el **buffer** de entradas por teclado está vacío.

### CLG |expresion|

Borra los gráficos de la pantalla dejando el color de fondo de éstos al determinado por **expresión**, o en su defecto, al actual.

```
10 PRINT "222"
20 DRAW 50,50
30 FOR x=1 TO 1000
40 CLG 15
50 PRINT "333"
60 DRAW 100,100
```

### CLOSEIN

Cierra la entrada de información procedente de un fichero en disco.

### CLOSEOUT

Cierra la salida de información hacia un fichero en disco.

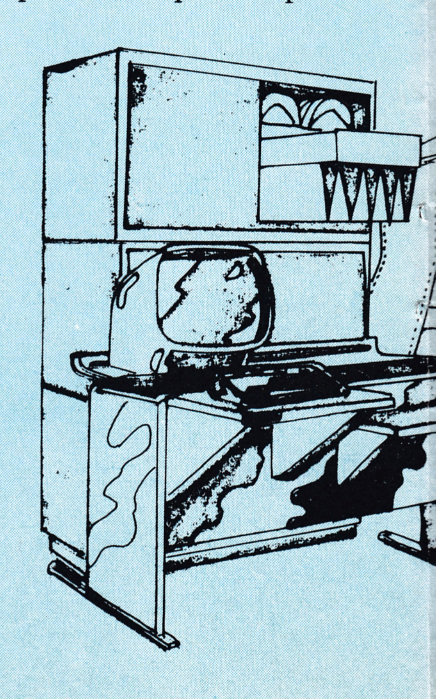
### CLS # expresion |

Borra la ventana indicada por # **expresión** a su color actual de fondo. La **expresión** debe dar de un número entre 0 y 7. Si # **expresión** se omite, se asume # 0, borrándose lo que en estos momentos se ve en pantalla.

```
10 PRINT "333"
20 WINDOW # 7,1,80,1,10 : PAPER # 7,15 :
   CLS # 7
30 PRINT # 7, "222"
40 FOR x=1 TO 3000 : NEXT
50 CLS
```

### CONT

Permite continuar la ejecución de un programa tras un BREAK producido por dos pul-



saciones sucesivas de ESC, o por un STOP en modo programa.

*COPYCHR$ ¡(#expresion)!*

Permite copiar el carácter que está situado en la actual posición del cursor en la ventana dada por **expresión.**

```
10 CLS : PRINT "HOLA" : LOCATE 4,1
20 WINDOW # 7,1,80,15,25
30 a$=COPYCHR$ (#0)
40 LOCATE #7,10,2
50 PRINT #7 a$
```

Con este programa se transfiere la A de HOLA, de la ventana #0, a la posición 10,2 de la ventana # 7.

*COS ¡(expresion)!*

Determina el coseno de la **expresión** en radianes (RAD) o grados (DEG), por defecto en radianes.

```
PRINT COS (PI/2)
0
```

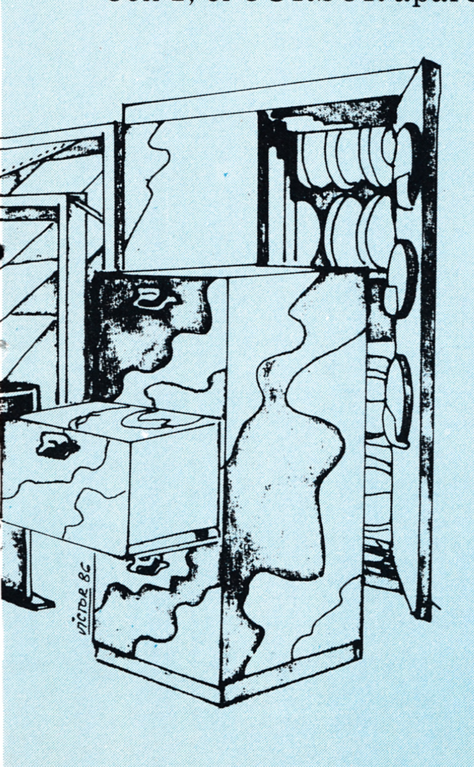
*CREAL ¡(expresion)!*

Se obtiene el número real correspondiente a la **expresión**

```
PRINT CINT (10/3) ,, CREAL (10/3)
3                3,33333333
```

*CURSOR ¡expresion!*

La **expresión** debe resultar un 1 o un 0. Con 1, el CURSOR aparece, con 0 desaparece.



```
10 x=1 : CURSOR x
20 PRINT INKEY$ ;
30 IF INKEY$="q" THEN CURSOR 0 :
STOP
40 GOTO 20
```

Con este programa, un cursor irá indicando la posición del próximo carácter a imprimir, hasta que se pulse la q.

*DATA ¡constante 1, constante 2,..., constante n!*

Cada **constante** puede ser numérica o alfanumérica. Almacena valores y los deja a disposición del comando READ.

Las constantes son leidas una tras otra por los READ de un programa.

DATA 255, JUAN, 1, AMSTRAD, 3, 127, 2

*DEC$ ¡expresion, "formato"!*

Convierte la **expresión** en una cadena representativa del número dado por ella y con el **formato** indicado.

Para los "especificadores" del formato, ver APENDICE A

```
PRINT DEC$ (2*PI*30, "###.####")
188.4956
```

*DEF FN ¡variable!(parámetros)¡=expresion!*

Permite al usuario definir sus propias funciones, expresión, con el nombre dado por variable, y pudiendo definir los **parámetros** implicados cuando sea necesario.

```
10 DEF FN A(x,y) = x+y*2
20 FOR x=1 TO 10
30 LET Z=FN A(x,10.1)
40 PRINT Z;
50 NEXT
21.2 22.2 23.2 24.2 25.2 26.2 27.2 28.2 29.2 30.2
```

*DEFINT ¡letra 1, letra 2,..., letra n!*
*o defint ¡letra 1-letra n!*

Convierte el valor de cualquier variable que comience con **letra 1, letra 2,** etc. o esté incluida en el campo determinado por **letra 1-letra n**, en el número entero más próximo.

Si al programa del ejemplo anterior se le añade:

```
5 DEFINT Z
```

La serie se transforma en

```
21 22 23 24 25 26 27 28 29 30
```

# Programas comentados

## CUENTOS FANTASTICOS

He aquí un programa, relatador incansable de cuentos fantásticos e irrepetibles.

Como todos los cuentos, comienza con el "Erase una vez..." pero la historia es cada vez distinta y su final impredecible.

Todo el programa gira en base a cinco fases básicas, que constituyen la estructura del cuento y que son complementadas con otras fases o palabras, elegidas aleatoriamente por el ordenador. A cada frase básica, le corresponden cinco o diez complementarias.

Estructura general del cuento:

1. ERASE UNA VEZ... (personaje del cuento)
2. QUE (acción que realizó)
3. SU (objeto sobre el que realizó la acción) (lugar en que la realizó).
4. Y COMO CONSECUENCIA (acción resultante) (lugar en que se produjo).
5. Y NUNCA MAS SE SUPO (frase final).

Como puede verse las frases 1, 2 y 5 tienen una frase complementaria, mientras que las 3 y 4, tienen dos.

Los cuentos o relatos resultantes son cortos, pero será muy fácil añadir nuevas frases básicas y complementarias, modificando el pgm. en el dimensionamiento de las matrices y añadiendo más frases y palabras en las líneas DATA.

Como sugerencia, te aconsejamos que tomes un libro o poema de todos conocidos y para cada frase básica construyas una serie de palabras complementarias, algunas de ellas disparatadas, de esta forma te aseguramos que ni el propio Cervantes sería capaz de reconocer su "Quijote", escrito por tu Amstrad.

***La zona izquierda del cuadro, constituye una matriz (array) de caracteres, unidimensional, formada por 5 elementos, que son las frases básicas.***

En la tabla siguiente se establecen las frases y correspondencia entre las frases y palabras usadas por el programa desarrollado.

La zona izquierda del cuadro, constituye una matriz (array) de caracteres, unidimensional, formada por 5 elementos, que son las frases básicas.

La zona derecha, forma una matriz bidimensional de caracteres, con 7 filas y 5 columnas, en la que cada elemento, representa una palabra o frase complementaria.

La estructura del programa es sencilla, una rutina de carga nos servirá, para introducir los textos, contenidos en líneas DATA, en las matrices correspondientes, debidamente dimensionadas.

A la primera matriz, que contiene las frases básicas, le daremos de dimensión cinco y por nombre F, como consecuencia la dimensionaremos cmo F$ (5). A la segunda, que nombraremos C, y que tiene 7 filas por 5 columnas, las dimensionaremos C$ (7,5).

A continuación presentamos el conjunto del pgm. comentado línea a línea, de manera que sea fácilmente comprensible y nos facilite el hacer modificaciones, como indicamos anteriormente.

70 Direccionamiento a subrutina de carga de frases.
80-100 Borrado de pantalla. Inicialización de A y B y actualización de la var. A (a, contiene el número de frase básica).
110 Control de fin de pgm., cuando el número de frases básicas se ha agotado, A 6).
120 Escribe frase básica.

| FRASES BASICAS | FRASES COMPLEMENTARIAS | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| ERASE UNA VEZ UN... | VIEJO | LAGARTO | NENE | VAMPIRO | BORRICO | 1 |
| QUE | ENVOLVIO | COMIO | DURMIO | PINTO | ROMPIO | 2 |
| SU | CABEZA | TAZA | PIE | NARIZ | CARRO | 3 |
| | EN EL WATER | EN LA CAMA | POR VOLVERSE VERDE | CON CHOCOLATE | CON ARENA | 4 |
| Y COMO CONSECUENCIA | SE PERDIO | APRENDIO BASIC | EXPLOTO | CRECIO MUCHO | ENGORDO | 5 |
| | EN UN RIO | COMO UN RAYO | EN LA COCINA | EN UN VASO | CON PIENSO | 6 |
| Y NUNCA MAS SE SUPO | COMO MURIO | DONDE SE ENCUENTRA | QUE SE LO LLEVO | COMO VOLO | CON QUIEN SE CASO | 7 |

# en BASIC

130 Actualiza var. B. (B contiene la fila de las complementarias).
140 Establece un valor aleatorio para la var. N, comprendido entre 1, 5, que corresponde a la columna de frases complementarias (INT RND X 5) genera números entre 0 y 4, por eso se le suma una unidad).
150 Escribe la frase complementaria correspondiente a la fila B, columna N (ver frases en el cuadro).
180 Direccionamiento control nuevo juego (380-440)
190-200 Borra líneas anterior de la pantalla y va al comienzo del juego (90), después de la rutina de carga.
210 Subrutina de carga de frases.
220 Dimensiona matriz de frases básicas.
230 Carga frases en la matriz.
240 DATA conteniendo frases básicas.
250 Dimensiona matriz frases complementarias.
260-290 Carga frases.
300-360 Datas frases complementarias.
370 Fin de subrutina y retorno al pgm. principal.
380-440 Subrutina de control para un nuevo relato, las líneas 400 a 410 constituyen un bucle indefinido, hasta tanto no se presione una tecla. Si la tecla pulsada es “C” se efectua el retorno al pgm., si es “P” fin de pgm. cualquier otra tecla, provoca la entrada al bucle de lectura del teclado (400-410).

```
10 REM  ..................
20 REM
30 REM     RELATOS FANTASTICOS
40 REM
50 REM
50 REM     ..................
60 REM
70 GOSUB 210
80 CLS
90 A=0: B=0
100 A=A+1
110 IF A=6 THEN GOTO 180
120 PRINT : PRINT L$(A);" ";
130 B=B+1
140 N=INT(RND*5)+1
150 PRINT P$(B,N);
160 IF B=3 OR B=5 THEN GOTO 130
170 GOTO 100
180 GOSUB 380
190 PRINT CHR$(11)+"
190 PRINT CHR$(11)+"               ":PRINT
200 GOTO 90
210 REM CARGA DE LAS MATRICES DE DATOS ................................
220 DIM L$(5) :REM FRASES BASICAS
230 FOR K=1 TO 5:READ L$(K): NEXT K
240 DATA " ERASE UNA VEZ UN...", "QUE", "SU", "COMO CONSECUENCIA," "Y NUNCA MAS SE SUPO"
250 DIM P$(7,5)
260 FOR F=1 TO 7
270 FOR C=1 TO 5
280 READ P$(F,C)
290 NEXT C,F
300 DATA"VIEJO","LAGARTO","VAMPIRO","BORRICO","JOVEN"
310 DATA"ENVOLVIO","COMIO","DURMIO","PINTO","ROMPIO"
320 DATA"CABEZA","TAZA","NARIZ", "MORRO","CARRO"
330 DATA " EN EL WATER"," EN LA CAMA"," AL VOLVERSE VERDE", "CON CHOCOLATE", " EN UNA NUBE"
340 DATA "SE PERDIO","APRENDIO BASIC", "EXPLOTO", "CRECIO MUCHO", "ENGORDO"
350 DATA " EN UN RIO"," COMO UN RAYO"," EN LA COCINA"," EN UN VASO"," CON PIENSO"
360 DATA "COMO MURIO","DONDE SE ENCUENTRA","QUE SE LO LLEVO", "PORQUE SALIO VOLANDO","CON QUIEN SE CASO"
370 RETURN
380 PRINT "=================================="
390 PRINT : PRINT "PRESIONA: <C> CONTINUAR, <P> PARAR"
400 Z$ = INKEY$
410 IF Z$="" THEN GOTO 400
420 IF Z$="C" OR z$="c" THEN RETURN
430 IF Z$="P" OR Z$="p" THEN END
440 GOTO 400
```

# INTRODUCCION A LOS

## CAPITULO I

POR JUAN MARTINEZ PINTOR

## INTRODUCCION

*LA palabra Logo deriva de la palabra latina "logos", que significa "pensamiento", "idea".

* Características del lenguaje:

— No requiere conocimientos previos de ninguna clase, ni tiene cota en cuanto a su potencial.

— Es un lenguaje estructurado. Los programas se crean combinando procedimientos.

— Es un lenguaje interactivo. La comunicación hombre/máquina, puede hacerse de forma inmediata a través de la consola.

— La estructura del lenguaje está muy relacionada con los objetos que maneja: números, palabras y listas.

— El Logo cuenta con los famosos gráficos tortuga. Esto facilita su aproximación con éxito a los niños.

— Es un lenguaje diseñado con objetivos educativos.

* Los mensajes de error los emite el ordenador cuando hacemos algo incorrectamente.

* Al trabajar en Logo, hay muchas palabras que el ordenador conoce: son los primitivos del sistema.

* Los conceptos de "palabra" y "lista", son muy importantes desde el punto de vista estructural.

* Una palabra es una serie de caracteres consecutivos, ninguno de los cuales puede ser un espacio en blanco, ni ningún otro separador de palabras.

* Una serie de palabras, perfectamente delimitada es una lista.

* Un "objeto-Logo", es una palabra o una lista.

* En general una lista es una serie de objetos-Logo perfectamente delimitada en su ordenación, comienzo y fin.

* Una manera de indicar el comienzo y el final de una lista, es mediante el uso de caracteres "[" y "]", respectivamente. Entre dos objetos consecutivos de una lista, solo media uno o varios espacios en blanco.

* Los programas Logo se llaman "procedimientos". Tienen un nombre que los identifica.

* Del mismo modo que una lista puede formar parte de otra, un procedimiento puede formar parte de otro.

### I.1. EL LOGO Y LOS LENGUAJES DE PROGRAMACION

Los ordenadores han inundado nuestros hogares y nuestras aulas. Del buen uso que se les dé en los ambientes educativos, dependerá en gran medida la formación de los hombres de las próximas generaciones. El lenguaje de programación Logo ha sido diseñado para acercar la informática a los niños del modo más natural y más rentable posible desde el punto de vista educativo.

A diferencia de las palabras FORTRAN, COBOL o BASIC, la palabra Logo no está

# PROGRAMAS EN LOGO

formada por siglas. Al contrario, deriva de la palabra latina logos, que a su vez proviene del vocablo griego que designa "idea", "pensamiento".

El lenguaje Logo ha sido desarrollado por el profesor Seymour Papert y su equipo, en el laboratorio de inteligencia artificial del Instituto Tecnológico de Massachussets (MIT), hace poco menos de 20 años. Papert es un matemático que ha dedicado mucho tiempo a investigar los procesos de aprendizaje y, por tanto, a pensar sobre el propio pensamiento, sobre las ideas. Por ello ha diseñado un lenguaje de programación de ordenadores con el que aprender a pensar. A diferencia de lo que se hace en otros ambientes, en el que el alumno sigue los pasos que le marca un programa (por ejemplo de ejercicios), en ambiente Logo, es el alumno el que dice al ordenador lo que debe hacer: el niño programando al ordenador en vez de lo contrario, según afirma el propio Papert.

En su libro **MINDSTORMS: Children, Computers and Powerful Ideas** (publicado en castellano con el título "Desafío a la mente: computadoras y educación", por Ediciones Galápago), Papert mantiene la tesis de que el Logo además de ser una modalidad para el uso de los ordenadores, es también un instrumento para explorar el conocimiento.

Para Papert son de la mayor importancia desde el punto de vista educativo, las ideas que el individuo ha hecho suyas, comprendiéndolas en profundidad y asimilándolas. Son lo que denomina los propios modelos. Para él, el hecho fundamental del aprendizaje (expuesto en el libro anterior), es éste: cualquier cosa es fácil si uno puede asimilarlo a la propia colección de modelos. Si eso no es posible, cualquier cosa puede resultar angustiosamente difícil. De modo que lo que un individuo' puede aprender y como lo aprende depende de los modelos con que cuenta.

De todo lo anterior resulta la importancia de situar lo antes posible a los niños en un ambiente en el que el propio pensamiento sea de la mayor importancia.

Al avanzar en el estudio del lenguaje, veremos como la estructura de sus programas posee un estilo diferente al de otros. Una estructuración que nos cautiva por su compleja sencillez.

Algunas características del lenguaje, son:

— Para comenzar a estudiar LOGO es suficiente un conocimiento básico de matemáticas. Incluso una persona que no posea conocimiento alguno podría comenzar a estudiar LOGO con éxito. Por otra parte, tampoco tiene una cota en cuanto a su potencial. De modo que las personas que quieran trabajar con LOGO, tienen abierto un gran campo.

— LOGO es un lenguaje **estructurado**. Los programas en este lenguaje se crean combinando estructuras de "rutinas", llamados "procedimientos", cada uno de los cuales podría estar construido del mismo modo. La importancia de este hecho, se irá viendo al ir progresando en el estudio del lenguaje.

— Es un lenguaje **interactivo**. Esto significa, que la comunicación hombre-máquina, puede hacerse de forma inmediata a través del teclado y la pantalla. Así, para ejecutar muchas de las órdenes del lenguaje y sus programas, basta escribirlas en el teclado.

— La forma en que LOGO maneja la información incluye "números", "palabras" y "listas". Además, la propia estructura del lenguaje está muy relacionada con estos objetos, lo que le hace particularmente capacitado para algunos tipos de problemas en los que haya que manejar símbolos.

— Una característica muy importante del LOGO, es el hecho de que posee los llamados "gráficos-tortuga". Los gráficos tortuga pre-

sentan muchas características interesantes. Por ejemplo, son un medio fabuloso para iniciar el estudio de la programación. Además, permiten al iniciado realizar estudios de matemáticas basadas en ordenador.

— De lo anterior deducimos ya que LOGO es un lenguaje diseñado también con objetivos educativos. En efecto, la facilidad de los gráficos tortuga permiten el acercamiento con éxito a los ordenadores incluso a los niños muy pequeños. Esto, ya de por sí muy interesante, unido a la estructura de sus programas, puede permitir que las nuevas generaciones de estudiantes, posean una organización mental de gran riqueza y, desde luego, completamente nueva. Con lenguaje de programación como LOGO (y sus sucesores), podremos conseguir que una mayoría de los niños se interesen de modo natural (e incluso amén), el propio razonamiento que les permite progresar en sus juegos y en sus hallazgos con el ordenador. Este amor por el pensar si será una revolución educativa. En este sentido, hemos de tomar al LOGO como un lenguaje más para aprender que para enseñar.

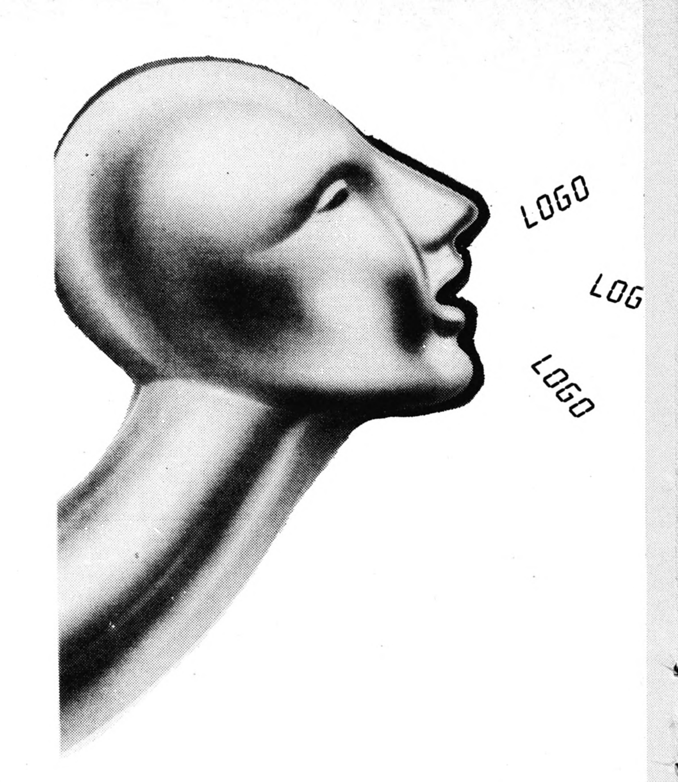
## *I.2. ESTRUCTURA DE LOS PROGRAMAS LOGO: PRIMITIVOS Y PROCEDIMIENTOS*

Una de las características del lenguaje Logo por las que más nos gustará trabajar con él, es porque es un lenguaje estructurado. Y por la estructura que tienen sus programas. A continuación veremos la primera aproximación a estas ideas.

Vamos a suponer que encendemos el ordenador, y que instalamos en su memoria el lenguaje Logo. En la ficha número 1, vemos como se hace esto. A modo de experimento escribamos la palabra "hola" (sin las comillas) y pulsemos después la tecla "enter". Entonces el AMSTRAD responde escribiendo en pantalla el mensaje.

**I don't know how to hola**

(no sé cómo se hace hola). Los mensajes de error los emite el ordenador como respuesta a




algo que hemos hecho incorrectamente. En otro capítulo volveremos sobre ellos.

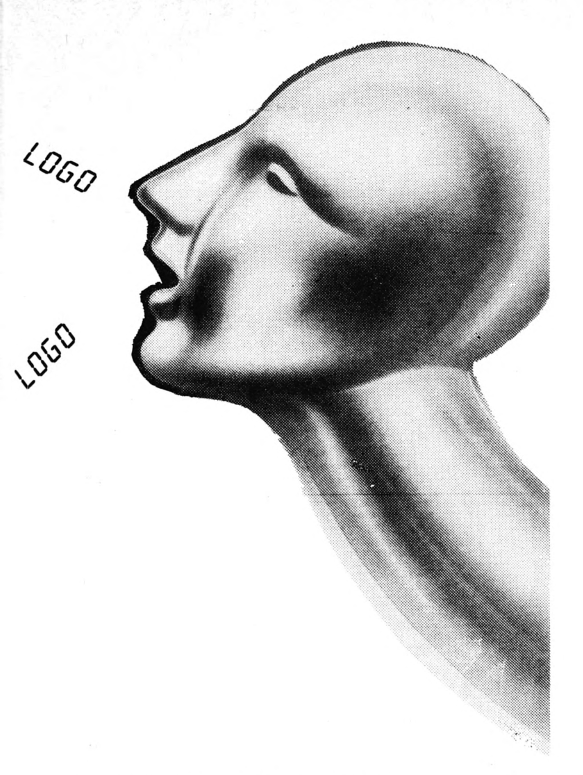
De modo que la palabra "hola" no la interpreta bien el ordenador; no la conoce. Escribamos ahora "ct" y pulsemos "enter". Vemos que la pantalla se borra, y no aparece ningún mensaje de error. De modo que la palabra **ct** sí la conoce el Logo: es una de sus instrucciones.

Todos los lenguajes manejan palabras como herramienta importante: es natural que los lenguajes estén formados así. Pero los conceptos de **PALABRA** y **LISTA**, son fundamentales desde el punto de vista estructural en Logo. Una palabra en Logo es una serie de caracteres consecutivos, ninguno de los cuales puede ser un espacio en blanco. El espacio en blanco es el separador de palabras. Hay algunos otros caracteres que tampoco pueden formar parte de las palabras, en general, porque son de algún modo también separadores de ellas. Tal es por ejemplo el caso del carácter +.

**EJEMPLO I.1**
— Son palabras **pa1,** pa.2, pa.3, **349.**
**— No son palabras np+1, np 5**.

Conforme vayamos avanzando en el estudio del Logo, iremos viendo qué caracteres no pueden formar parte de una palabra.

Como vimos antes, al cargar el Logo hay una serie de palabras que conoce, y por lo




tanto con ellas podemos ordenarle determinadas cosas. Estas palabras son las que en Logo se llaman **PRIMITIVOS**.

Naturalmente, los programas llevarán a efecto acciones mucho más complicadas que las desarrolladas por los primitivos. Antes de referirnos a la estructura de los programas, veremos el concepto de lista, al que ya hemos hecho referencia, y que necesitaremos más adelante.

El concepto de lista es muy potente y rico, entre otras muchas cosas porque subyace a toda la estructura del lenguaje. Una serie de palabras, con indicación de donde empieza y donde acaba, es una lista en Logo. En general un **objeto-Logo** es una palabra o una lista. Pues bien, resulta que una lista es una serie de elementos cada uno de los cuales es un objeto Logo, es decir, una palabra o una lista. De este modo, podemos tener listas de varios niveles formando parte, como objetos-Logo de una lista; y palabras al mismo nivel que listas... Una manera (no la única), de indicar a Logo que una serie de objetos ha de ser tomada como una lista, es encerrando sus elementos entre los caracteres "[" y "]".

### EJEMPLO 1.2

Como pa1, pa2 y pa3 son palabras, una lista formada por ellas en ese orden será: [pa1 pa2 pa3].

Como pa4 es otra palabra, otra lista más compleja que incluya como elementos palabras y listas será: [[pa1 pa2 pa3] pa4]. Esta lista está formada por dos objetos-Logo: la lista [pa1 pa2 pa3] y la palabra pa4.

Observemos que los distintos objetos-Logo de una lista, sólo se separan entre sí por espacios en blanco.

¿Da lo mismo el orden en que se sitúen los diferentes objetos de una lista? **NO**. La lista anterior y la lista [pa4 [pa1 pa2 pa3]], son distintas.

Quizás estemos ya intuyendo por donde van las cosas desde el punto de vista estructural, en la confección de programas en Logo. Un pequeño programa podría muy bien estar formado solamente por una serie de órdenes elementales (primitivos), del mismo modo que una lista puede estar formada sólo por palabras. Pero tal vez queramos más adelante hacer un programa un poco más complejo, en el cual el anterior sólo representa una parte. Podemos imaginar que los programas tengan un nombre, y que nos referimos a ellos precisamente por su nombre: una palabra. Entonces esa palabra (que no es ahora un primitivo), está formada por otras, por ejemplo primitivos... ¡sí, del mismo modo que una lista debajo nivel está formada por palabras! Pero el programa 'un poco mayor' que utiliza a varios del tipo de nuestro programita se parece entonces a una lista que estuviera formada por varias listas y, tal vez algunas palabras.

Vemos pues, que aunque el ordenador al empezar a trabajar no sabía cómo hacer "hola", es posible definir un programa con ese nombre y asi enseñar al ordenador cómo debe hacer esa orden.

Esta es en efecto la estructura del lenguaje. Los programas complejos se dividen en partes, que realizan acciones un poco más simples, cada una de las cuales a su vez se dividen posiblemente en nuevas partes... en niveles arbitrarios de complejidad. Son lo que se llaman **PROCEDIMIENTOS** en Logo. Del mismo modo que una lista puede formar parte de otra, un procedimiento puede formar parte de otro. En realidad, más adelante, cuando estemos ya trabajando con el LOGO del ordenador AMSTRAD, realizaremos el proceso de incluir unos procedimientos en otros, casi sin darnos cuenta.

# DOCTOR LOGO

## Programas comentados en LOGO




ES indispensable en programación que el estudio teórico sistemático de todas las instrucciones y detalles del lenguaje, vaya complementado con la práctica continuada casi desde el primer momento.

En estos "ejercicios" el lector interesado podrá profundizar en la práctica de la programación. En efecto, presentaremos los listados de muchos y variados programas escritos en el lenguaje Logo y a su derecha se comentarán las instrucciones y procedimientos. Cuando sea necesario, se incluirán recuadros en los que se aclaren ideas más complejas o largas y en los que se desarrollen y aclaren conceptos de otras materias que sea necesario utilizar. El lector no interesado en esos detalles podrá pasarlos por alto sin más.

Cada programa irá acompañado también por unas instrucciones de uso. Esto permitirá al usuario interesado plantearse el propio programa como un ejercicio e intentar desarrollarlo el mismo. Posteriormente se puede consultar el listado aquí propuesto y comparar los detalles. Además, nada impide tampoco que se hagan nuevas versiones mejoradas de los programas propuestos.

En el extremo opuesto, las personas cuyo interés esté sólo en utilizar los programas pueden ignorar la parte de listado y comentarios, concentrándose directamente en la máquina. No es nuestro consejo. Aparecerán, aquí programas de todos los tipos: desde sencillas utilidades que nos ayudarán posteriormente a trabajar en los nuestros, hasta sofisticados programas (no necesariamente de juegos). Para ayudar a elegir la manera en que debemos afrontar cada programa, estos llevarán al lado de su nombre un número de 'D': una sola D para indicar que el programa es fácil, dos para indicar que es medio y tres para los considerados difíciles. El criterio adoptado se refiere, naturalmente, a personas que no dominan el lenguaje.

Para teclear los programas escritos en Logo hemos de situar el ordenador en este lenguaje. Encendamos pues la máquina, introduzcamos el disco que contiene el Dr. Logo y escribamos |CPM. Después de algunos segundos la máquina está dispuesta para trabajar en Logo. En un primer paso, y mientras avanzamos en nuestro estudio del Logo, lo mejor es teclear los programas tal y como

aparezcan en el listado. Cuando queramos guardar en un disco todos los programas que tenemos en la memoria, buscamos un nombre (por ejemplo, **nombre**) y entonces escribimos: save "nombre". Para cargar este trabajo posteriormente en la memoria, desde el disco que lo contiene, se escribe: load "nombre".

## MINIMAX (LOGO) (Procedimientos con resultado) (D)

**Instrucciones de manejo:**

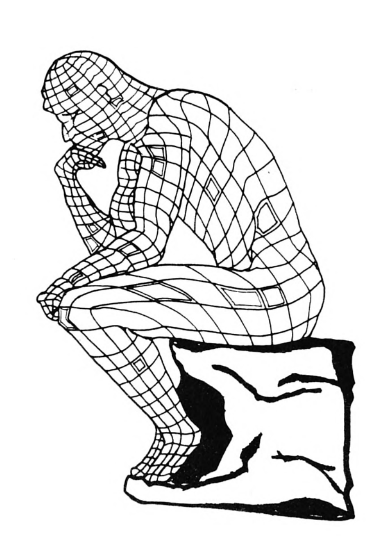
El programa "MINIMAX" contiene dos utilidades. Una para obtener el mínimo de dos números y otra para obtener el máximo.

El procedimiento para hallar el mínimo se llama "min"; requiere dos números como datos de entrada y da como resultado el menor de ellos.

De igual modo el procedimiento "max" da como resultado el mayor de sus dos datos de entrada.

Como es usual en Logo, estos datos de entrada no han de ser números necesariamente, sino que pueden ser procedimientos que den números como resultado. Asimismo, es posible concatenar cuantos procedimientos deseemos, sin más que tener en cuenta que las prioridades son las usuales: primero se ejecutan los procedimientos más internos y progresivamente los más externos.

POR JUAN MARTINEZ PINTOR



EJEMPLOS:

```
?pr min 2 7
2
?pr max 3+4 23
23
?pr max 3 min 7 1
3
?pr 2*max .5 1.5
3.
```



| LISTADO | COMENTARIOS |
|---|---|
| to min :a :b | La primera palabra es el nombre del procedimiento. Las otras son para los datos. |
| if :a<:b [op :a] | La primera parte es el condicional "si :a es menor que :b". Si la condición se cumple se ejecuta la lista encerrada entre corchetes. |
| op :b | La orden "op :a" hace que el procedimiento de como resultado el valor de :a y termine ahí su ejecución. |
| end | |
| to max :a :b<br>if :a< :b [op :b]<br>op :a | Como la anterior. Pero sólo una de ellas se ejecuta. |
| end | Fin del procedimiento. |




SELLO

**Bailén, 20 - 1.º Izq.**

**28005 - MADRID**



# COMENTARIOS

## POLIG (LOGO
## (La estructura repetitiva) (D)

**Instrucciones de manejo:**

El programa "POLIG" consiste en dos utilidades. Permite dibujar, con la tortuga del Dr. Logo, polígonos regulares. Estos polígonos pueden tener el número de lados y la longitud de lados que se decida al ejecutar los procedimientos.

El programa consiste en dos procedimientos, para dibujar los polígonos hacia la izquierda o hacia la derecha, de la posición de la tortuga. Los nombres respectivamente son "polii" y "polid".

Por ejemplo, para dibujar un polígono regular de tres lados (triángulo) de modo que la tortuga gire hacia la izquierda y de modo que cada lado mida 100 pasos de tortuga, basta ejecutar "polii 100 3". Del mismo modo, "polid 140 8" dibuja un octogono de lado 140 y de forma que la tortuga gira hacia la derecha.

Este tipo de utilidades es bueno almacenarlas en los discos para que podamos utilizarlas en cualquier momento, sin más que cargarlas en la memoria.

LISTADO

```
to polii :lado :número

repeat :número [fd :lado

It 360/número]
end

to polid :lado :número
repeat :número [fd :lado
  lt 360/ :número]
end
```

COMENTARIOS

El nombre del procedimiento es la primera palabra. Las otras dos palabras son los nombres de las variables locales que utiliza.

Al ejecutar el procedimiento las variables locales adquieren un valor. Se repetirá la lista de instrucciones encerrada entre los corchetes, el número indicado de veces.

# El programa técnico del mes

## Las crecidas en los ríos y arroyos. Cálculo de caudales de aven

### INTRODUCCION

EL paso de caudales extraordinariamente grandes, por un río, arroyo o barranco, se denomina avenida, riada o crecida.

Sus efectos negativos son tan conocidos como indeseados, por ello el estudio de los fenómenos que producen las riadas, es de gran interés, así como la evaluación de las mismas, que nos permitirá, el tratar de protegernos de sus efectos.

¿Qué causas provocan las riadas? Generalmente, salvo casos bastante infrecuentes, como la rotura de una presa de embalse o similares, hay dos fenómenos naturales que pueden provocar riadas: las lluvias intensas y la fusión rápida de la nieve. Una combinación de ambos fenómenos también es posible, y de hecho sus efectos se potencian al aparecer simultáneamente.

En nuestra geografía, sin ser despreciables, los efectos de la nieve, son menos importantes que los productos por las lluvias copiosas, dándose con relativa frecuencia lluvias muy intensas en corto espacio de tiempo y en zonas donde el de precipitaciones anuales es muy reducido.

Este artículo girará en torno a la lluvia como origen de las avenidas.

### GENERALIDADES SOBRE EL FENOMENO

Las gotas de agua desprendidas por un aguacero, siguen al caer al suelo, trayectorias muy distintas: Una parte se deposita en la parte superior del suelo, dándole humedad, que será absorbida por la vegetación y el resto se perderá por evaporación. Otra parte es retenida directamente por las plantas y otra se depositará formando charcos, que posteriormente se evaporarán.

Todos estos fenómenos y caminos que toma el agua, corresponden a precipitaciones de poca cuantía, pero en aguaceros importantes, a medida que el suelo se va saturando de agua, va dejando escurrir, cada vez, una mayor cantidad de agua caída y simultáneamente se va produciendo una infiltración, en el subsuelo, pasando a engrosar las aguas subterráneas.

Estas aguas subterráneas, acaban desaguando en los cauces, pero en este proceso invierten generalmente un largo tiempo.

A diferencia de las anteriores las aguas superficiales, desaguan con gran rapidez en los cauces, y son evacuadas en un corto espacio de tiempo. Como consecuencia, puede afirmarse que es la escorrentía superficial, el determinante principal de la formación de avenidas, por lo que al menos en una primera aproximación, puede prescindirse de los demás factores.

De la totalidad del agua caída, como hemos visto, sólo una parte da lugar a escorrentía. Este volumen de agua se conoce como "lluvia neta".

Ahora bien, existe un retraso entre la entrada del agua en la cuenca, y su salida por el punto de desagüe, debido al tiempo invertido por las aguas en su trayecto, la medida de este retraso se denomina, "Tiempo de concentración", que no depende de la cantidad de agua, que constituye la escorrentía, sino únicamente de las características físicas de la cuenca, longitud y pendiente del terreno.

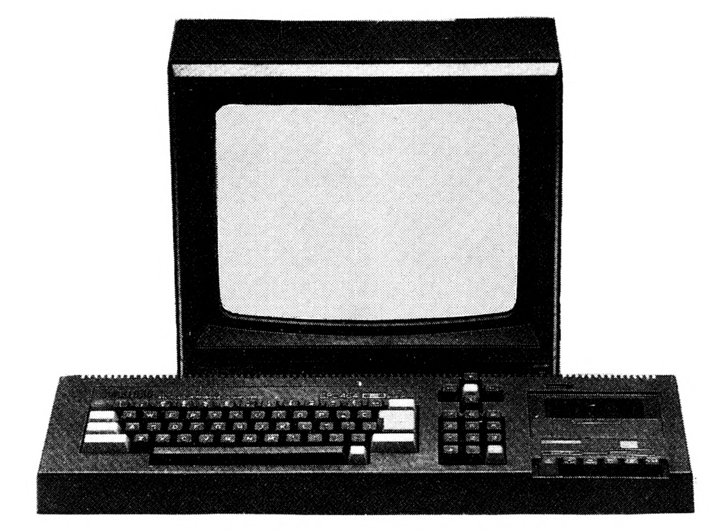
Este y otros conceptos hidrológicos, son necesarios para el análisis de la problemática de la escorrentía, pero no es nuestro interés, el realizar un estudio teórico completo, sino el facilitar un método rápido para obtener un valor, suficientemente aproximado, de la magnitud del caudal máximo previsible de avenida, con una cierta garantía.

### METODOLOGIA UTILIZADA EN EL PROGRAMA

El programa que presentamos, responde al criterio expuesto anteriormente, de sencillez, rapidez y suficiente aproximación.

De entre los numerosos estudios existentes, sobre el cálculo de caudales de avenida, el programa está basado en el estudio expuesto en la

# da en pequeñas cuencas naturales



VICTOR J. CAMPO LOPEZ

obra "Cálculo Hidrometeorológico de Caudales Máximos en Pequeñas Cuencas Naturales" de don José Ramón Temez, doctor Ingeniero de C. C. y P. (Dirección General de Carreteras M.O.P.U.). Este estudio está basado estructuralmente en el "Método Racional", que reune los requisitos de sencillez y fiabilidad. Los parámetros que intervienen en esta ley general, han sido adaptados al caso de España Peninsular y contrastados experimentalmente.

## UTILIDAD DEL PROGRAMA

El programa que presentamos a continuación, permite obtener el caudal máximo previsible en una cuenca, para un período de retorno dado, en función de los parámetros de la cuenca y de la intensidad de lluvia máxima previsible.

El conocimiento de este caudal nos permitirá dimensionar con suficiente garantía, aquellas obras que hayan de realizarse sobre el cauce o proximidades, alcantarillas, pontones, desvios, azudes, tomas de aguas, etc.

El programa, se aplicará únicamente a cuencas, cuya extensión no sea superior a 75 Km² y que estén ubicadas, dentro de la Península.

**FORMULACION EMPLEADA.** Para aquellos que desean profundizar en el programa, exponemos la formulación básica utilizada.

— Tiempo de concentración (Tc)

0.76

$$Tc = 0.3 \times \frac{L^{0,76}}{J^{1/4}}$$

L = Longitud del curso principal
J = Pendiente media

— Intensidad-Duración.

$$\frac{I}{Id} = \frac{Il}{Id} \; \frac{28^{0.1} - D^{0.1}}{0.4} \qquad \frac{Il}{Id} = \text{Relación intensidad horaria/diaria}$$

D = Tiempo

— Precipitación máxima diaria (Pd).

Obtenida a partir de los mapas de Isoyetas (ver ejemplo).

— Coeficiente de escorrentía.

$$C = \frac{(Pd\text{-}Po)\,(Pd+23\,Po)}{(Pd+11\,Po)^2}$$

Po = Parámetro propio de la cuenca, ver su determinación en "Uso del pgm"

— Caudal Máximo

C.I.A.

$$Q = \frac{C.I.A.}{3}$$

Fórmula Racional
A = Area de la cuenca I = Intensidad

## GLOSARIO DE TERMINOS UTILIZADOS

A continuación exponemos una relación de palabras, utilizadas en este artículo, no con el fin de ser tenidas en cuenta como definición, sino aclaratorias, para aquellas personas con pocos conocimientos de Hidrología.

— Caudal: Cantidad de agua que fluye por una sección, en la unidad de tiempo.

— Lluvia neta: Parte de la lluvia caída, que produce la escorrentía.

— Escorrentía: Agua que fluye sobre la superficie del terreno o bajo él. (Esc. superficial. o subterránea).

— Cuenca: Area receptora de aguas y conductora de las mismas, a un cauce común.

— Cauce: Parte más profunda de la cuenca, por donde fluyen las aguas.

— Tiempo de concentración: Medida del efecto de retraso en la escorrentía, debido al trayecto que debe recorrer el agua, desde los puntos de caída hasta el punto de concentración considerado.

— Parámetro Po: Se produce escorrentía, cuando la suma de la lluvia acumulada desde el comienzo del aguacero es igual a Po.

— Avenida, riada, crecida: Paso por un cauce, de caudales extraordinariamente grandes.

— Divisoria: Línea límite que separa cuencas adyacentes. Se obtiene uniendo los puntos más altos, que delimitan la cuenca.

## USO DEL PROGRAMA

### * ENTRADA DE DATOS

Introduzca los valores correspondientes a los INPUTS, que se relacionan a continuación, en orden de aparición en pantalla.

— Nombre de la cuenca: Nombre del arroyo, río o barranco. Si no se conoce pulsar ENTER.

— Superficie de la cuenca en $Km^2$: Si no se conoce a priori, puede obtenerse planimetrando la cuenca, en la cartografía disponible. Tenga en cuenta que el pgm. sólo admite superficies menores de 75 $Km^2$, por tanto si rebasa esta cifra, se producirá un mensaje de aviso y el pgm. volverá al principio.

— Longitud máxima del cauce Km: Esta longitud corresponde al cauce principal (más largo). Si se desconoce puede obtenerse midiendo sobre la misma cartografía en que se determinó el área.

— Pendiente media: en m/m: Es la pendiente media del cauce principal. Puede obtenerse dividiendo la diferencia entre la cota más alta del cauce y la más baja, por la longitud. Si el cauce es muy irregular, puede dividirse por tramos y calcular la pendiente media ponderada de dichos tramos.

— Il/Id: Esta relación puede obtenerla del gráfico que el ordenador le presenta en pantalla, interpolando entre las curvas del mismo según la situación geográfica de la cuenca.

— Máxima precipitación en 24 horas: Puede obtenerse (si se desconoce) con suficiente aproximación de los planos de Isoyetas, publicados por la Dirección General de Carreteras (M.O.P.U.), para el período de retorno considerado. (Puede consultarse La Instrucción de Carreteras Precipitaciones max. anuales en 24 horas).

— Período de retorno: Introduzca el valor en años, de acuerdo con la max. precipitación considerada.

— Valor medio de Po: La estimación de Po (parámetro de umbral de escorrentía), se realiza de acuerdo con las características del terreno y tipo de cultivo, mediante las tablas que exponemos a continuación.

## TABLA PARA LA ESTIMACION INICIAL DEL PARAMETRO Po

| Uso de la tierra | Pendiente % | Características hidrológicos | Grupo de suelo A | B | C | D |
|---|---|---|---|---|---|---|
| Barbecho | ⩾ 3 | R | 15 | 9 | 6 | 4 |
| | | N | 17 | 11 | 9 | 6 |
| | < 3 | R/N | 20 | 14 | 11 | 8 |
| Cultivos en hilera | ⩾ 3 | R | 23 | 13 | 8 | 6 |
| | | N | 25 | 16 | 11 | 8 |
| | < 3 | R/N | 28 | 19 | 14 | 11 |
| Cereales de Invierno | ⩾ 3 | R | 29 | 17 | 10 | 8 |
| | | N | 32 | 19 | 12 | 10 |
| | < 3 | R/N | 34 | 21 | 14 | 12 |
| Relación de cultivos pobres | ⩾ 3 | R | 26 | 15 | 9 | 6 |
| | | N | 28 | 17 | 11 | 8 |
| | < 3 | R/N | 30 | 19 | 13 | 10 |
| Relación de cultivos densos | ⩾ 3 | R | 37 | 20 | 12 | 9 |
| | | N | 42 | 23 | 14 | 11 |
| | < 3 | R/N | 47 | 25 | 16 | 13 |
| Praderas | ⩾ 3 | Pobre | 24 | 14 | 8 | 6 |
| | | Medio | 53 | 23 | 14 | 9 |
| | | Buena | — | 33 | 10 | 13 |
| | | Muy buena | — | 41 | 22 | 15 |
| | < 3 | Pobre | 58 | 23 | 12 | 7 |
| | | Media | — | 35 | 17 | 10 |
| | | Buena | — | — | 22 | 14 |
| | | Muy buena | — | — | 22 | 14 |
| | | Muy buena | — | — | 25 | 13 |
| Plantaciones regulares de aprovechamiento forestal | ⩾ 3 | Pobre | 62 | 26 | 15 | 10 |
| | | Media | — | 34 | 19 | 14 |
| | | Buena | — | 42 | 22 | 15 |
| | < 3 | Pobre | — | 34 | 19 | 14 |
| | | Media | — | 42 | 22 | 15 |
| | | Buena | — | 30 | 25 | 16 |
| Mares forestales (bosques, monte bajo, etc.) | | Muy claro | 40 | 17 | 8 | 5 |
| | | Clara | 60 | 24 | 14 | 10 |
| | | Media | — | 34 | 22 | 16 |
| | | Espesa | — | 47 | 31 | 23 |
| | | Muy espesa | — | 65 | 43 | 33 |
| Rocas permeables | ⩾ 3 | | 3 | | | |
| | < 3 | | 5 | | | |
| Rocas impermeables | ⩾ 3 | | 2 | | | |
| | < 3 | | | | | |

LABORES DE CULTIVO

En línea recta (símbolo R).

Cuando el laboreo del suelo, la siembra y las labores de cultivo se realizan en la dirección de la máxima pendiente o a media ladera.

En líneas de nivel (símbolo N).

Cuando el laboreo del suelo, la siembra y las labores de cultivo se realizan siguiendo las curvas de nivel del terreno. Evidentemente en terrenos llanos no resulta fácil, ni tiene mucho sentido, matizar las líneas de nivel, por lo que no se hace diferencia entre el laboreo en línea recta (R) y el laboreo en línea de nivel (N).

SUELOS

Grupo A

En ellos el agua se infiltra rápidamente aún cuando estén muy húmedos. Profundos y de texturas gruesas (arenosas o areno-limosas), están excesivamente drenados.

Grupo B

Cuando están muy húmedos tienen una capacidad de infiltración moderada. La profundidad de suelo es de media a profunda, y su textura franco-arenosa, franca, franco-arcillo-arenosa o franco-limosa según terminología del U. S. Department of Agriculture. Están bien o moderadamente drenados.

Grupo C

Cuando están muy húmedos la infiltración es lenta. La profundidad de suelo es inferior a la media y su textura es franco-arcillosa, franco-arcillo-limosa, limosa o arcillo-arenosa. Son suelos imperfectamente drenados.

Grupo D

Cuando están muy húmedos la infiltración es muy lenta. Tienen horizontes de arcilla en la superficie o próximos a ella y están pobremente o muy pobremente drenados. También se incluyen aquí los terrenos con nivel freático permanentemente alto y suelos de poco espesor (litosuelos).

Si la cuenca es bastante uniforme, introduzca el valor obtenido de la tabla. Si no lo es (diversidad de cultivos, suelos etc.) divida la cuenca en zonas homogéneas, halle la superficie de cada zona, calcule con la tabla el Po correspondiente y tome para el pgm. el valor medio ponderado de Po.

— Multiplicador regional: Este parámetro trta de corregir el valor de Po, de acuerdo con el grado de humedad existente, antes de comenzar el aguacero y se obtiene directamente interpolando en el gráfico que se ofrece por pantalla, de acuerdo con la situación geográfica de la cuenca dentro de la Península.

## * SALIDA DE RESULTADOS

El pgm. presenta un cuadro con todos los valores físicos de la cuenca y parámetros introducidos anteriormente y el valor del caudal máximo previsible de avenida, para el período de retorno considerdo.

EJEMPLO DE APLICACION: Sobre el barranco de los Regueros, en Fresno de Torote, Madrid, se ha de ubicar una obra de paso, para un camino vecinal, en el punto que se indica en el plano.

Se ha delimitado la cuenca trazando la línea divisoria, hasta el punto de ubicación de la obra.

Se desea conocer el caudal máximo previsible, para un período de retorno de 10 años, con el fin de dimensionar la alcantarilla de paso.

Planimetrando la cuenca, se obtuvo una superficie de 2.18 $Km^2$. La longitud del cauce principal es de 2.35 Km.

La pendiente media será:

| | |
|---|---|
| cota más alta | 792 |
| cota más baja | 730 |
| desnivel | 62 m. |

$$J = \frac{\text{desnivel}}{\text{longitud}} = \frac{62}{2350} = 0.026$$

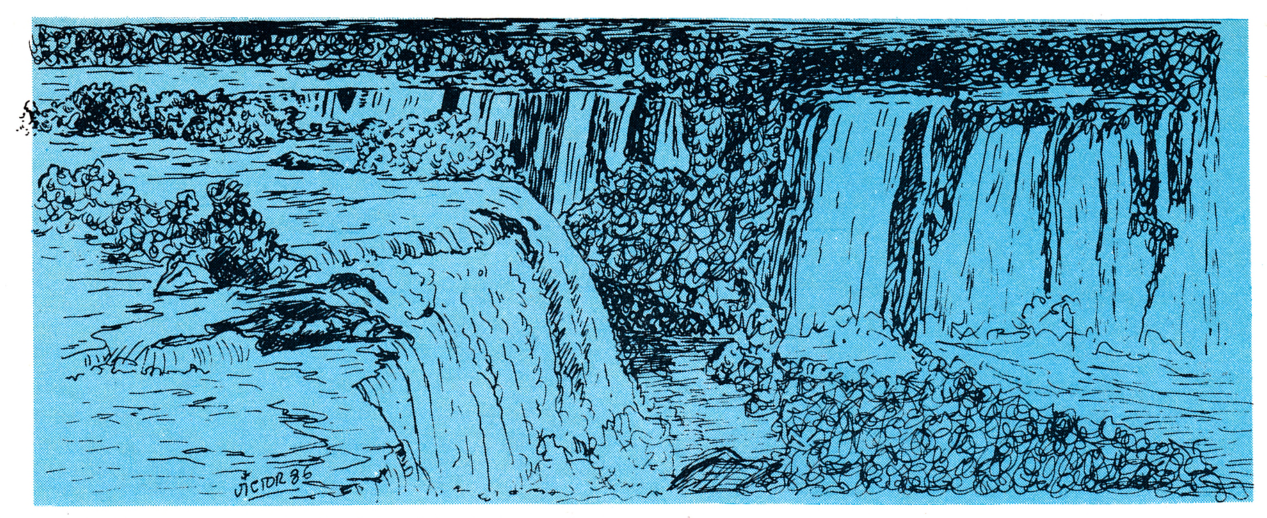
Del gráfico de pantalla, obtenemos un valor de Il/Id = 9, de acuerdo con la zona de ubicación de la cuenca.

La precipitación máxima en 24 horas. Se ha obtenido del plano de "Precipitaciones máximas anuales en 24 horas", para PR = 10 años, y es 90 mm. El período de retorno considerado es 10 años.

El valor medio de la Po se ha obtenido dividiendo la cuenca en dos zonas, una de monte bajo que representa el 70% de la superficie y la otra con rocas impermeables.

| | Po |
|---|---|
| Zona 1) Monte bajo, muy claro, con suelo tipo C ............................ | 5 |
| Zona 2) Suelo formado por rocas impermeables .......................... | 8 |

$$\text{Po medio} = \frac{5 \times 30 + 8 \times 70}{100} = 7.1$$




El multiplicador regional obtenido interpolando en el gráfico de pantalla, es 2.8.

Una vez introducido todos los datos en el programa, se obtiene como caudal máximo previsible 8.92 $m^3/s$., para el período de retorno de 10 años considerado.

```
10 REM ................
20 REM
30 REM. CALCULO DE CAUDALES DE .
40 REM. AVENIDA EN PEQUENNAS .
50 REM. CUENCAS NATURALES .
60 REM.
70 REM ...............
71 MODE 0: PRINT CHR$(24): PRINT "      ";
   "CAUDALES DE AVENIDA";
72 PRINT "         "; "EN CUENCAS NATU-
   RALES";
73 PRINT "_______" : PRINT CHR$(24)
74 LOCATE 7,15:PRINT "AMSTRAD":LOCA-
   TE 15,16: PRINT "Club"
75 FOR K=1 TO 7000:NEXT K
80 MODE 1: LOCATE 10,5: PRINT "DATOS
   DE LA CUENCA":PRINT TAB(9);"_______":
   PRINT
90 INPUT "NOMBRE DE LA CUENCA";C$
100 INPUT "Superficie cuenca (Km2)";A
110 GOSUB 520
120 INPUT "Longitud max. cauce (Km) ";]
130 INPUT "Pendiente media (m/m)       ";j
140 CLS : GOSUB 570
150 GOSUB 820
160 LOCATE 1,23
170 INPUT "I1/ID ";I1
180 TC=0.3*(L/J 0.25) 0.76
190 IID=I1 ((28 0.1-TC 0.1)/0.4)
200 INPUT "MAX. PRECIPITACION (mm/
    24h.) ";PD
205 INPUT "PERIODO DE RETORNO";PR
210 ID=PD/24: I=IID*ID
220 INPUT "VALOR MEDIO DE PO";PO
230 CLS: GOSUB 570
240 GOSUB 2040
249 LOCATE 1,21
250 INPUT "MULTIPLICADOR REGIONAL"
    ;MR
260 PO=PO*MR
270 C=(PD-PO)*(PD+23*PO)/(PD+11*PO) 2
280 Q=C*I*A/3 : REM fórmula racional
290 f2$= "PtPtPt.PtPt":F5$="PtPtPt.PtPtPtPtPt"
300 MODE 1
310 WINDOW Pt1,1,17,5,15
320 WINDOW Pt2,21,40,5,15
330 PRINT Pt1," S="; USING F2$;A;: PRINTPt1,
    "Km2"
340 PRINT Pt2,"J ="; USING F5$;J;: PRINTPt2,
    "m/m"
350 PRINT Pt1, " L="; USING F2$;L;: PRINT
    Pt1, "Km"
360 PRINT Pt2, "I/="; USING F2$;I1
370 PRINT Pt1, " TC="; USING F2$;TC;: PRINT
    Pt1, "A."
380 PRINT Pt2,"MR="; USING F2$;MR
390 PRINT Pt1, "Po="; USING F2$;PO/MR;:
    PRINTPt1," mm."
400 PRINT Pt2,"PD="; USING F2$;PD;: PRINT
    Pt2, "m/m"
410 PRINT TAB(10); "CUENCA - ";C$
420 PRINT TAB(5); "PERIODO DE RETORNO
    ";PR;" A."
425 PRINT CHR$(24)
430 LOCATE 9,12: PRINT "CAUDAL =";USING
    "PtPtPtPtPt.PtPt";Q;
431 PRINT "M3/s."
440 LOCATE 1,22
450 PLOT 0,350:DRAW 600,350
460 PLOT 0,250:DRAW 600,250
470 PLOT 0,185:DRAW 600,185
480 PLOT 0,185:DRAW 0,350
490 PLOT 295,250:DRAW 295,350
500 PLOT 600,185:DRAW 600,350
505 PRINT CHR$(24)
510 END: REM FIN DE PROGRAMA ======
    ==============================
    ===
520 IF A <=75 THEN RETURN
530 LOCATE 1,16
540 PRINT "EL METODO SOLO ES VALIDO
    PARA CUENCAS MENORES DE 75 Km2
550 FOR K=1 TO 6000:NEXT K
560 GOTO 80
570 REM ............ DIBUJO MAPA PE-
    NINSULA .......
580 MODE 2
610 RESTORE 730
```

Próximo programa del mes: "SISTEMAS DE UNIDADES".




# AMSTRAD
# CPC 464, 664 y 6128
# PROGRAMACION ESTRUCTURADA

Cuando este libro salía para la imprenta el Amstrad lanzaba el micro CPC 6128 doméstico. Sus ventajas sobre el 664 es que está equipado con el doble de memoria de usuario RAM, un teclado diferente (que a mi gusto no es tan bueno como el del 664) y una nueva versión del sistema operativo industrial CP/M que le permite funcionar con un surtido más amplio de programas de gestión.

La buena nueva es que los programas escritos en BASIC para el 464 y el 664 funcionarán sin alteración en el 6128. Eso significa que debieran «currar» todos los programas basados en el disco 664 y el 99,5% de los programas en cinca 464. **Eso significa también que puedes aprovechar todas las pistas y trucos de programación de este libro para aprender a usar tu nuevo 6128.**

Aunque el 6128 se suministra con 128K de RAM, sólo 68K están disponibles para los programas en BASIC. Eso es porque el BASIC del 6128 se diseñó originalmente para el 464 y el 664 que sólo tienen disponibles 64K. Para ayudarte a solventar este problema, Amstrad ha incluido algunos comandos BASIC extra que pueden implantarse desde el disco. Con ellos se te permite usar el «repuesto» de 64K de RAM bien sea para almacenar imágenes de pantalla adicionales o bien como sistema rápido de archivo. Por el momento, no puedes usar esos 64K de repuesto para contener programas en BASIC.

Todos estos comandos extra quedan implantados en el sistema haciendo que se ejecute el programa en código máquina «Bankmanager» suministrado en tu disquete. Si quieres utilizar esa RAM de repuesto para almacenar en ella imágenes de pantalla, el Bankmanager (gestor de las bancadas de memoria) te proporciona dos comandos adicionales.

| SCREENCOPY y | SCREENSWAP que te permiten conservar hasta 4 imágenes de pantalla en la memoria adicional y luego **canjearlas** sobre la pantalla. Eso puede ser útil para juegos y animación en los que se prepara la nueva pantalla en la «retaguardia» y luego se pasa a «vanguardia» cuando es necesario. Cada imagen de pantalla se identifica con un número de bloque del 1 al 5. Para que se proyecte en el monitor es necesario moverla al bloque 1.

AMSTRAD
CPC 464, 664 Y 6128
PROGRAMACION ESTRUCTURADA
STEPHEN RAVEN
Comience
Camine hasta la parada del autobus
Espere hasta que llegue su autobus
Llega un autobus
Sí
no
Súbase al autobus
Otra ronda de acciones
Finalice
ra-ma

| SCREENCOPY hace que se **copie** toda una imagen de pantalla desde un bloque origen hasta un bloque destino. El contenido previo del bloque destino se pierde siempre al escribir encima. Por ejemplo | SCREENSWAP, 1, 4 pondría la imagen corriente en el bloque 4 y el contenido del bloque 4 pasaría a ser el proyectado en pantalla. Un ulterior | SCREENSWAP haría que las imágenes retornaran a la situación primitiva.

Si decides que quieres usar el repuesto de 64K de RAM para almacenar datos en lugar de imágenes en pantalla el «gestor de bancada» te proporciona 4 comandos BASIC adicionales:

| BANKOPEN permite que se **abra** la bancada especificando cuantos caracteres habrá en cada registro. La máxima longitud es de 255 caracteres. | BANKWRITE hace que se **escriba** un registro, | BANKREAD te permite la **lectura** de un registro y | BANKFIND hace que se **halle** un registro que concuerde con un literal dado y devuelve el número identificativo del registro de manera que puedas efectuar una lectura posterior para conocer los datos registrados.

Todos los comandos aparte del de apertura de bancada necesitan especificar una variable entera que en ella se contenga el «código de estado» y una variable literal para reseñar los datos cuya lectura o escritura se va efectuar.

**P.V.P. 1.700 ptas.**

# NUESTRO PRÓXIMO NÚMERO



# AMSTRAD EDUCATIVO N.º 1

## Fe de Erratas

*Pág. 27:*

— *Donde pone*

*LISTADO*

*to polii :lado : número*

*repeat: número (fd :lado*

*lt 360/ número)*
*end*

*to polid: lado :número*
*repeat :número (fd: lado*
*lt 360/ :número)*
*end*

— *Debe poner*

*LISTADO*

*to polii : lado :número*

*repeat :número (fd : lado*

*Lt 360/:número)*
*end*

*to polid :lado :número*
*repeat : número (fd: lado*
*rt 360/ :número)*
*end*

*Página 29:*

— *Donde pone:*

— *Tiempo de concentración (Tc)*

$$Tc = 0.3 \times \frac{L^{0,76}}{J^{1/4}}$$

*L = Longitud del curso principal*
*J = Pendiente media*

— *Intensidad - Duración*

— *Debe poner:*

— *Tiempo de concentración (Tc)*

$$Tc = 0.3 \left(\frac{L}{J^{1/4}}\right)^{0.76}$$

*L = Longitud del curso principal*
*J = Pendiente media*

— *Intensidad - Duración*

## LISTADO DEL PROGRAMA TECNICO DEL MES

## Las crecidas en los ríos y arroyos. Cálculo de caudales de avenida en pequeñas cuencas naturales.

---

**Nota.**—El símbolo Pt se debe sustituir por #

```
10 REM ............................
20 REM .                            .
30 REM .   CALCULO DE CAUDALES DE   .
40 REM .   AVENIDA EN PEQUENNAS     .
50 REM .   CUENCAS NATURALES        .
60 REM .                            .
70 REM ............................
71 MODE 0: PRINT CHR$(24): PRINT "                          " ;"CAUDALES DE AVENIDA ";
72 PRINT "                     "; "EN CUENCAS NATURALES";
73 PRINT "__________________" : PRINT CHR$(24)
```

```
74 LOCATE 7,15: PRINT  "AMSTRAD":LOCATE 15,16:  PRINT  "Club"
75  FOR K=1 TO 7000:NEXT K
80 MODE 1: LOCATE 10,5: PRINT "DATOS DE LA CUENCA":PRINT TAB(9);"-------------------"; PRINT
90 INPUT  "NOMBRE DE LA CUENCA ";C$
100  INPUT  "Superficie  cuenca (Km2) ";A
110  GOSUB 520
120  INPUT  "Longitud max. cauce (Km) ";l
130  INPUT  "Pendiente media (m/m)    ";j
140 CLS :  GOSUB 570
150  GOSUB 820
160 LOCATE 1,23
170  INPUT  "I1/ID ";I1
180 TC=0.3*(L/J^0.25)^0.76
190 IID=I1^((28^0.1-TC^0.1)/0.4)
200 INPUT "MAX.PRECIPITACION (mm/24h.) ";PD
205 INPUT  "PERIODO DE RETORNO";PR
210 ID=PD/24: I=IID*ID
220 INPUT  "VALOR MEDIO DE PO ";PO
230 CLS:  GOSUB 570
240  GOSUB 2040
249 LOCATE 1,21
250 INPUT  "MULTIPLICADOR REGIONAL ";MR
260 PO=PO*MR
270 C=(PD-PO)*(PD+23*PO)/(PD+11*PO)^2
280 Q=C*I*A/3 : REM formula racional
290 f2$= "###.##";F5$="###.#####"
300 MODE 1
310 WINDOW #1,1,17,5,15
320 WINDOW #2,21,40,5,15
330 PRINT #1," S ="; USING F2$;A;: PRINT#1," Km2"
340 PRINT #2,"J ="; USING F5$;J;: PRINT#2, " m/m"
350 PRINT #1," L ="; USING F2$;L;: PRINT#1," Km"
360 PRINT #2,"I/="; USING F2$;I1
370 PRINT #1," TC="; USING F2$;TC;: PRINT#1," A."
380 PRINT #2,"MR="; USING F2$;MR
390 PRINT #1," Po="; USING F2$;PO/MR;: PRINT#1," mm."
400 PRINT #2,"PD="; USING F2$;PD;: PRINT#2, " m/m"
410  PRINT TAB(10); "CUENCA -  ";C$
420  PRINT TAB(5); "PERIODO DE RETORNO ";PR;" A."
425 PRINT  CHR$(24)
430 LOCATE 9,12: PRINT "CAUDAL =";USING "#####.##";Q;
431 PRINT  " M3/s."
440 LOCATE 1,22
450 PLOT 0,350:DRAW 600,350
460 PLOT 0,250:DRAW 600,250
470 PLOT 0,185:DRAW 600,185
480 PLOT 0,185:DRAW 0.350
490 PLOT 295,250:DRAW 295,350
500 PLOT 600,185:DRAW 600,350
505 PRINT  CHR$(24)
510 END: REM F I N  D E  P R O G R A M A======================================
520 IF A<=75 THEN RETURN
530  LOCATE 1,16
540 PRINT  "EL METODO SOLO ES VALIDO PARA CUENCAS   MENORES DE 75 Km2
550 FOR K=1 TO 6000:NEXT K
560 GOTO 80
```

```
570 REM ........................................ DIBUJO MAPA PENINSULA .......
580 MODE 2
610 RESTORE 730
620  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
630 FOR K=1 TO 48: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
640 NEXT K
650 RESTORE 780
660  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
670 FOR K=1 TO 7: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
680 NEXT K
690 RESTORE 790
700  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
710 FOR K=1 TO 4: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y
720 NEXT K
730 DATA 190,168,186,160,163,161,152,159,136,163,134,165,130,163,108,169,102,165,106,161
740 DATA 93,161,89,157,90,152,94,153,92,147,97,145,94,139,94,119,89,99,82,87
750 DATA 82,80,88,84,86,77,90,77,90,65,86,51,99,51,103,49,110,53,119,52
760 DATA 133,32,139,32,141,38,150,40,152,44,184,44,196,59,206,61,204,64,208,74
770 DATA 217,80,212,82,208,92,222,113,222,118,252,137,250,147,248,151,248,156
780 DATA 94,139,104,139,109,135,124,136,129,128,121,121,112,74,110,53
790 DATA 186,160,216,149,221,153,238,146,250,147
800 DATA 108.142,127,159,159,159,171,157,178,142,182,158,187,150,218,130,238,130,150,130,150,112,158,98,109,54,130,55,166,
        50,198,66, 206,92
810 RETURN
820 REM ........................................ ISOLINEAS I1/ID ............
825 LOCATE 1,7:  PRINT CHR$(24)+ "---------------------"
826 LOCATE 1,8:  PRINT  "A MAPA DE ISOLINEAS A"
827 LOCATE 1,9:  PRINT "---------------------"
828  PRINT CHR$(24)
830 RESTORE 850: z=3:  GOSUB 2000
840 LOCATE  44,1: PRINT  8
850 DATA 152,165,160,157,188,159,192,166
860 z=2:  GOSUB 2000
870 DATA 110,170,119,153,208,155
880 LOCATE  32,1: PRINT  9
890 z=9:  GOSUB 2000
900 DATA 136,145,148,148,163,135.164,129,153,117,136,110,128,114,125,121,130,138,136,145
910 LOCATE  36,5: PRINT  10
920 z=7:  GOSUB 2000
930 DATA 142,142,157,135,154,124,138,113,132,116,131,127,136,139,142,142
940 LOCATE  38,7: PRINT  11
950 z=10:  GOSUB 2000
960 DATA 163,115,171,115,172,111,168,100,169,88,165,80,160,77,155,83,158,93,151,105,163,115
970 LOCATE  47.14: PRINT  "9"
971 LOCATE  48,10: PRINT  "M"
980 z=15:  GOSUB 2000
995 DATA 142,36,145,49,151.50,162,47,180,58,178.105,187,115,176,123,182,132,169,142,168,146,172,150,187,151,212,145,235,
        145,251,151
996 LOCATE 40,22: PRINT  "10"
1000 z=9:  GOSUB 2000
1010 DATA 190,48,198,78,195,96,202,115,191,125,187,134,172,145,177,148,216,134,254,145
1020 LOCATE 55,20: PRINT  "11"
1030 z=5:  GOSUB 2000
1040 DATA 212,72,202,100,206,115,204,128,219,130,234,135,252,133
1050 LOCATE 62,16: PRINT  "12"
1500 RETURN: REM Fin rutina Isolineas
```

```
2000 REM .... RUTINA DIBUJO .....
2010  READ X,Y:X=X*2.3: Y=Y*2.3: PLOT X,Y
2020 FOR K=1 TO z: READ X,Y:X=X*2.3: Y=Y*2.3: DRAW X,Y: NEXT K
2030 RETURN :REM Fin rutina dibujo
2040 REM ....................................... MULTIPLICADOR REGIONAL ......
2045 RESTORE 2060
2050 z=5:  GOSUB 2000
2060 DATA 106,135,120,141,132,156,148,158,186,156,196,160
2070 z=6:  GOSUB 2000
2080 DATA 112,88,128,88,168,98,190,110,201,135,232,137,254,133
2085 Z=5: GOSUB 2000
2090 DATA 140,36,144,47,180,52,184,72,192,80,218,84
2100 LOCATE 34,4 : PRINT  "2"
2110 LOCATE  46,11: PRINT  "3"
2120 LOCATE  48,10: PRINT  "M"
2130 LOCATE 53,14 : PRINT  "4"
2140 LOCATE 9,8: PRINT  CHR$(24)+" MAPA "
2150 LOCATE 1,9: PRINT  " MULTIPLICADOR REGIONAL ": PRINT  CHR$(24)
2170 RETURN
```

# EDITORIAL

*Muchas gracias por el interés que estáis mostrando por nuestra revista, a través de vuestras cartas, opiniones y sugerencias.*

*Estamos ya tabulando estas indicaciones para irlas adaptando a nuestro esquema, y la planificación se irá ajustando, de esta forma, a nuestras necesidades.*

*Los programas irán gradualmente haciéndose más complicados y difíciles, a medida que vayáis dominando los niveles más elementales.*

*Si tenéis en mente algún programa que os gustase ver desarrollado, hacédnoslo saber y nos pondremos manos a la obra inmediatamente para que, cuanto antes, lo veáis publicado en vuestra revista.*

*El objetivo es que nuestro apelativo "Educativo" sea cada vez más eso, una plataforma que te permita dominar, cada día más a fondo, las posibilidades de tu aparato, formando un triángulo entre la revista, tu aparato y tú.*

*Esperamos vuestras opiniones.*

# SUMARIO






**Edita:** Grupo Editorial G.T.S., S. A.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02.
**Director:** Antonio Bellido.
**Colaboran** en este número:
Juan M. Pintor, Víctor J. Campo López.
**Maquetación:** Susana M. Villalba.
**Secretaria de Redacción:** Mercedes Jamart.
**Publicidad:** Dpto. propio.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02

**Fotocomposición:**
Fotocom, S. A.
**Fotomecánica:**
La Unión
**Imprime:**
Gráficas FUTURA Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23
Políg. Ind. Cobo Calleja
FUENLABRADA (Madrid)
**Distribuye:**
R.B.A. Promotora de Ediciones, S. A.
Travesera de Gracia, 56 Atico, 1.ª.
Tfno. 93 - 200 82 56
**Depósito Legal:**
M. 8.904-1986

# EL AMSTRAD Y EL CP/M

## *Ejercicios con estructura CP/M*

1.° —¿Qué subgrupo de programas CP/M se encarga del directorio del disco en uso?

.........................................

Respuesta.— BDOS.

2.° —¿Qué proceso sigue el BDOS para dar entrada a cualquier fichero? .......

Respuesta 1. Comprueba en la pista catálogo la no existencia de otro fichero con el mismo nombre.

2. Comprueba si la capacidad libre del disco es suficiente para el nuevo fichero.

3. Superadas las dos comprobaciones anteriores, da entrada en la pista catálogo a los datos correspondientes al fichero.

3.° —Para transferir información desde una unidad de disco a una impresora:

¿Qué sección del DOS se encarga de comprobar si la impresora está ocupada?

.........................................

Respuesta.— BIOS.

¿Qué sección del DOS se encarga de localizar la información en el disco?

Respuesta.— BDOS

¿Qué sección del DOS se encarga de que la transferencia se efectúe con eficacia? ..........................

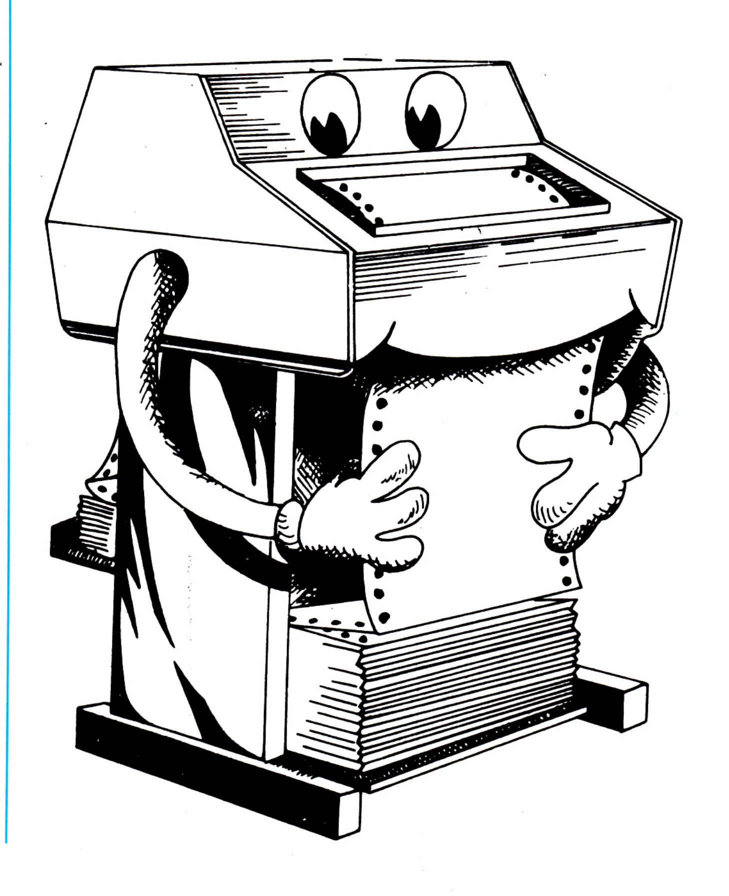
Respuesta.— BIOS.

4.° —¿Cuál es el acrónimo de la sección de CP/M encargada de procesar las órdenes del operador? ¿Qué significa en español? ......................

Respuesta.— CCP. Procesador de órdenes de consola.

5.° —¿Cómo se llaman las órdenes CP/M contenidas en el CCP? .............

Respuesta.— Permanentes o residentes.



## *Edición de línea*

Una de las acepciones del Diccionario para la palabra edición es "impresión o estampación de una obra o escrito para su publicación".

Aplicando esta idea al terreno sobre el que nos estamos moviendo, debemos interpretar el concepto de "edición de línea" como la acción que debe llevar a cabo el operador para imprimir en pantalla la orden a ejecutar por el DOS.

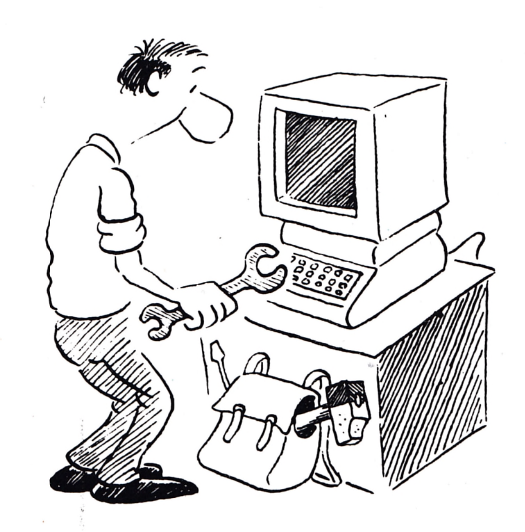
Esta acción pasa por escribir la orden correctamente en el teclado, y finaliza en el momento de pulsar la tecla de retorno de carro normalmente conocida por ENTER o RETURN. A esta tecla, y el efecto subsecuente que se produce al pulsarla, nos referiremos mediante <cr>.

Las órdenes en CP/M, dadas a través del teclado, pueden ocupar una línea de pantalla o más, pero, en todo caso, no pueden exceder de un número predeterminado de caracteres. Al conjunto de caracteres que configuran la orden se le da el nombre de "línea de orden", las líneas de orden dependiendo del ordenador, suelen permitir hasta un máximo de caracteres, comprendidos entre 127 y 255. Por tanto, debe consultarse el manual en este sentido.

Con toda seguridad, el operador, en algún momento, cometerá errores durante el tecleado o escritura de las líneas de orden y, por tanto, deberá conocer los mecanismos que CP/M pone a su disposición para corregirlos.

A estos mecanismos se les conoce con el nombre de **"órdenes de edición de línea".** Resumen de estas órdenes:

- **E.—** Cambia cursor al comienzo de la línea siguiente.
- **H.—** Borra último carácter.
- **J.—** Da por concluida la línea actual.
- **M.—** Retorno de carro.
- **R.—** Repite línea en edición.
- **S.—** "Congela" pantalla.
- **U.—** Cancela línea de orden actual.
- **C.—** Reinicializa el sistema.
- **P.—** Conmuta impresora.
- **Z.—** Da por terminado el proceso de entradas actual.



## *Ordenes de edición de línea*

El usuario puede modificar una línea de orden en cualquier momento antes de pulsar <cr>.

CP/M permite estas correcciones mediante ciertas órdenes de edición de línea, los cuales suelen representarse con una letra y con el signo asu izquierda. PC: E.

La acción que lleva a efecto cada orden de edición de línea se obtiene pulsando simultáneamente la tecla de control y la de la letra del comando correspondiente. En algunas máquinas la tecla ALT sustituye a la de control.

La función que lleva a efecto cada orden está controlada por el CCP.

Veamos, a continuación, un resumen:

**E**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL (o ALT según el ordenador) y la tecla de la letra E.

**Función asignada:** Producir un final físico en la línea que actualmente se está escribiendo. Al pulsar E, el cursor salta al comienzo de la línea inmediatamente inferior, pero la orden no pasa a la CPU para su ejecución. Util para línea de orden mayores que una línea de caracteres en pantalla. En todo caso, la línea de orden no se da por concluida hasta que no se pulsa <cr>.

**Ejemplo:** Supongamos que se desea dar la orden DIR, lo cual exigiría teclear D, I y R, y pulsar <c r>, apareciendo en pantalla

A > DIR

y a continuación, como veremos al estudiar esta orden, una relación de los ficheros que contiene el disco que esté instalado en la correspondiente unidad. Si con el fin de comprender las implicaciones de E escribiéramos, tras A>, cada una de las letras de DIR seguidas de E, obtendríamos la siguiente imagen

A > D
I
R

En estas condiciones, al pulsar <c r>, el resultado sería el mismo que en el supuesto primero.

De todo esto podemos inferir que E no afecta a la orden que se esté escribiendo, y sí, exclusivamente, a la forma en que se configura la misma en pantalla.

**H**

**Función asignada:** Mover el cursor atrás una posición de carácter borrando. Esta función se puede usar en tanto no se pulse <c r>.

**Formas de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra H. También apretando BACKSPACE, ROBUOT o DELETE si el ordenador dispone en su teclado de alguna de estas teclas.

**Ejemplo:** Si al teclear la orden DIR escribimos:

A > D I T < ■

y antes de dar por concluida la orden con <c r>, se utiliza H, por alguno de los procedimientos explicados, la T desaparece bajo el cursor (■) pudiendo pulsar, ahora, la R para conseguir la orden correcta.

**J**

**Formas de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra J o apretando LINE FEED (LF) si el teclado tiene esta función.

**Función asignada:** Dar por concluida la línea de orden y pasa a la siguiente: Equivale a <cr>. Su denominación es *Line feed* o alimentación de línea.

**Ejemplo:** La orden dada en los dos ejemplos de los comandos anteriores, se han concluido con un <c r>. Si cambiamos el <c r> por J el efecto sería el mismo.

**M**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra M o apretando <c r>.

**Función asignada:** Su denominación es *carriage return* o retorno de carro: obliga al cursor a volver al comienzo de líneas (margen izquierdo).

**Ejemplo:** Vale lo dicho en el anterior comando.

**R**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra R.

**Función asignada:** Volver a imprimir en pantalla la actual línea de orden, siempre, claro está, que no se haya concluido con <c r>. La misión encomendada a R no tiene mayor validez en CP/M 2.2 y 86, y se justifica en versiones anteriores ya que al borrar caracteres producían una doble impresión. Cuando se aplica R, la línea en pantalla se cancela con el símbolo #.

**Ejemplo:** Supongamos que, a título de curiosidad, tecleamos

A > DIR

y, sin pulsar <cr>, damos un R, la pantalla nos mostraría

A > DIR #
DIR

Con lo cual se consigue repetir lo escrito. Nada más.

**S**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra S.

**Función asignada:** Detener la impresión en pantalla de la secuencia de salida que, a la sazón, se esté produciendo.

Pulsando cualquier tecla el proceso continúa a partir del carácter donde se produjo la interrupción. Si una impresora está activada, con P como veremos, ésta responde de la misma forma.

**Ejemplo:** Tras una orden del tipo DIR, ya utilizada en otros ejemplos, la pantalla comenzará a mostrar los ficheros que actualmente tiene un determinado disco. En estas condiciones, si ordenamos S el listado se detendrá indefinidamente hasta que se pulse una tecla.

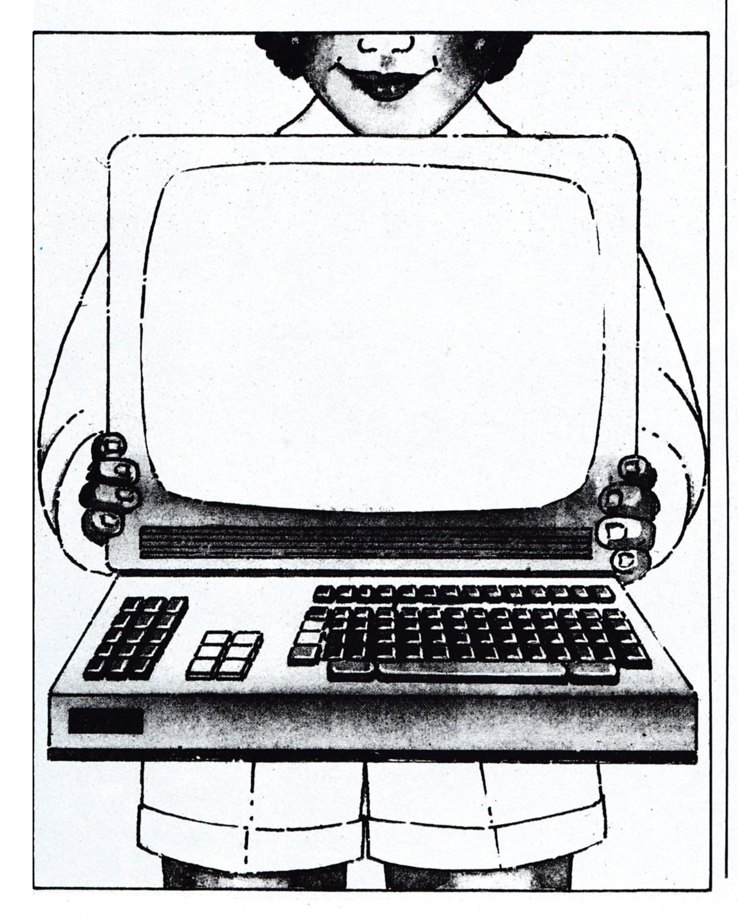
### U (X)

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra U (o de la letra X).

**Función asignada:** Anular la presente línea de orden, permitiendo comenzar a escribir una nueva sin que la anulada sea ejecutada. La línea en pantalla se cancela con el símbolo #.

**Ejemplo:** Al intentar dar la orden DIR, se ha escrito por equivocación:

A > DOM



Gracias a U, podemos anular la orden y comenzar de nuevo, siendo la situación resultante en pantalla:

A > DOM #

El cursor está ahora en condiciones de comenzar una nueva línea de orden.

**Nota:** U y X cumplen la misma función.

Hasta aquí los comandos de edición de línea que permiten manipular lo escrito en una orden antes de ser ejecutadas.

Veamos ahora otra serie de comandos que, aunque no actúan sobre una línea de orden en el sentido hasta ahora expuesto, sí permiten al operador imponerse al sistema.

### C

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra C.

**Función asignada:** Reinicializar el sistema y:

1. Catalogar el disco insertado en la unidad de disco que esté en uso, guardando en la RAM un mapa de asignaciones para el directorio (o catálogo) de dicho disco. Este proceso es conocido como **log in** o **logs.**

2. Interrumpir el proceso que se esté llevando a efecto para devolver el control al operador.

En CP/M 86, lo anterior es válido teniendo en cuenta que antes, y también mediante C, se deben interrumpir, uno tras otro, todos los procesos que se estén llevando a efecto.

En CP/M, y en términos generales, se debe dar C siempre que se cambien discos cuando el sistema está a nivel de comandos. Si se intenta escribir en el disco recién cambiado sin utilizar C, se obtendrá un mensaje de error.

**Ejemplo:** Habrá ocasión, más adelante, de probar este comando, no obstante, podemos tomar contacto mediante estos supuestos:

1. Si con la impronta A > en pantalla y un disco insertado en la unidad de disco, ordenamos un C, observaremos como el disco se pone en movimiento para ser recatalogado.

2. Si en el ejemplo propuesto en S, pulsamos C en lugar de S, se notaría que además de interrumpirse el listado, se devuelve el control al operador, mostrando la pantalla la impronta A >.

**P**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra P sin activar la salida por impresora. Repitiendo esta operación se desactiva.

**Función asignada:** Enviar todas las salida enviadas a la pantalla, también a la impresora. Ordenando nuevamente P se cancela la salida por impresora. Se debe tener precaución con P ya que, si la impresora no está debidamente conectada, el sistema se "colgará" y deberemos inicializar el DOS, perdiendo el contenido actual de la RAM.

**Ejemplo:** Activar la impresora con P y poner en práctica el ejemplo dado en S, con lo cual veríamos el listado salir por la impresora.

**Z**

**Forma de obtenerlo:** Pulsando simultáneamente CONTROL y la tecla de la letra Z.

**Función asignada:** Dar por terminada la entrada de información a través del teclado.

**Ejemplo:** Este comando se verá en acción en toda su plenitud, al estudiar las órdenes PIP y ED.

Sabiendo ya como escribir, y corregir si ha lugar, las líneas de orden, sigamos el camino que nos conduce a comprenderlas.

## *Caracteres de control*

1.° —¿Qué caracter
1.° —¿Qué carácter de control reinicializa el sistema? ......................

Respuesta.— C

2.° —¿Qué carácter de control congela la salida de información? .............

Respuesta.— S

3.° —¿Qué carácter lo descongela? .......

Respuesta.— Cualquiera, excepto C que reinicializa el sistema.

4.° —¿Qué carácter de control permite o prohibe, la salida de información a través de la impresora? ...........

Respuesta.— P

5.° —¿Qué carácter de control borra la línea de orden completa? ...............

Respuesta.— U y X

6.° —¿Qué carácter de control mueve el cursor una posición atrás? .........

Respuesta.— H

7.° —¿Qué misión tiene I? ...............

Respuesta.— Hace saltar el cursor al comienzo de la próxima zona de tabulación.



# Los ficheros en el AMSTRAD

## Cómo crear ficheros en disco

### *ORDENADOR - COMPUTADOR*

Según el diccionario de la Real Academia de la Lengua Española, **computar** significa **contar o calcular una cosa por números**. Y un **computador electrónico es un aparato electrónico que realiza operaciones matemáticas y lógicas con gran rapidez.**

En otros dicionarios se pueden encontrar definiciones en este sentido, pero más ampliadas:

"Computador: Ingenio que, por medio de circuitos lógicos, puede efectuar cualquier operación basada en la de contar, y que opera en el sistema binario de numeración y bajo los criterios del álgebra de Boole".

Así pues, un computador se limita a actuar de acuerdo con la definición anterior, y, consiguientemente, está incapacitado para actuar como una máquina de escribir, o un tubo de rayos catódicos... Pero si puede estar en condiciones de controlarlos y vincularse con ellos.

En este orden de cosas, es evidente que, si el computador no estuviera habilitado para comunicarse con el exterior, su utilidad sería nula, y, de aquí, la necesidad de unirlo con otros ingenios a través de los cuales pueda recibir mensajes del exterior (entradas) y, a su vez, emitir los suyos propios (salidas).

Concluyendo, un **computador** es una unidad física destinada a operaciones basadas en la de contar, y que opera en el sistema binario y bajo los criterios del álgebra de Boole, y un **ordenador** es una suma de distintas unidades físicas, cada una dedicada a su propia actividad y, todas ellas, dirigidas por un computador teniendo por fin el proceso de datos automáticamente.

Para completar estar puntualizaciones, necesarias por otra parte para salir del confusionismo existente y poder expresarnos con precisión, diremos que un sistema de computación equivale al concepto de ordenador.

### *CPU Y MEMORIA INTERNA*

De todos los elementos que componen un moderno computador, sólo hay dos que realmente interesen ahora a nuestro propósito: La CPU y la memoria.

La CPU, acrónimo de **Central Processing Unit**, vale por unidad central de proceso o microprocesador.

En el microprocesador se realizan todas las operaciones de control, comparación y aritméticas que tienen lugar en un computador; es un circuito integrado de alta densidad.

La memoria de un computador, imprescindible para que el microprocesador cumpla su función, está formada por una serie de **chips**.

Una disposición física real de la distribución de estos elementos sobre el circuito impreso principal, no nos ayudará tanto como el siguiente esquema, en el cual las flechas nos indican los flujos de información y los bloques de las partes fundamentales a considerar.

Unas palabras sobre la forma de instruir al microprocesador cierran este epígrafe.

Un microprocesador sólo puede procesar instrucciones y datos en forma de números binarios.

El conjunto de instrucciones que interpreta un microprocesador recibe el nombre de código máquina de ese determinado microprocesador.

Así, los microprocesadores 8080 A, 8085 y Z80 tienen un juego de instrucciones común, y, consiguientemente, pueden ejecutar programas desarrollados con el mismo código máquina.

A los microprocesadores 8086 y 8088 les sucede igual entre ellos.

## MEMORIA

Desde cualquier punto de vista, la memoria es la capacidad de retener y recordar información.

Según los especialistas, la memoria humana puede dividirse en tres sistemas principales: **memoria sensorial** a **corto plazo** y a **largo plazo.**

El almacenamiento sensorial de información es sumamente breve, llegando, en la visión, a una décima de segundo.

Haga la prueba de recordar el tacto de la última cosa que tocó y comprobará que o bien no puede, o bien le cuesta trabajo.

La memoria a corto plazo conlleva un mayor tiempo de retención de la información. Tal sería el caso de las reuniones de trabajo o las citas para comer en una fecha más o menos próxima.

La memoria a largo plazo se refiere a ese tipo de información que nuestro cerebro retiene constantemente. Un ejemplo típico y extremo podría ser la tabla de sumar que aprendimos de niños en el colegio.

Para concluir con esta breve incursión en terrenos que me son extraños, añadiré que una persona con buena memoria será, normalmente, más brillante y sufrirá menos inconvenientes que otra que tenga mala memoria.

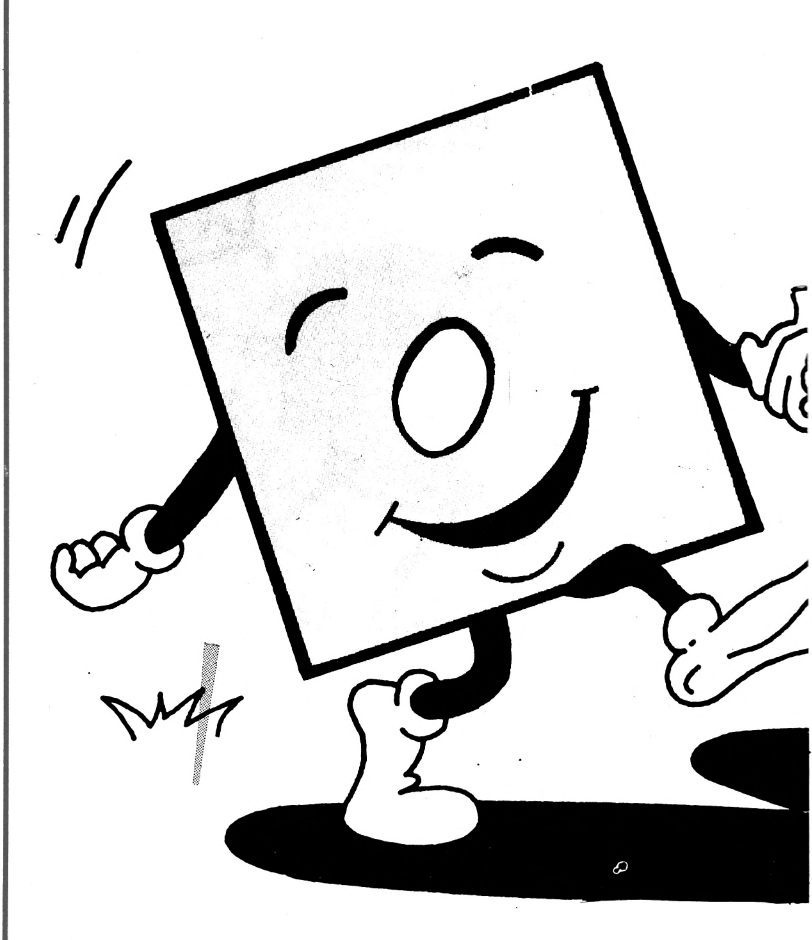
Veremos ahora la analogía existente entre lo expuesto anteriormente y la memoria de un ordenador.

Al trabajar en modo directo o inmediato a través del teclado de un ordenador, estamos utilizando una especie de memoria sensorial del computador, ya que, ejecutada la orden (al apretar ENTER), nada ha quedado en la memoria de éste.

Cuando utilizamos nuestra máquina como una calculadora, estamos aprovechando este tipo de memoria. El siguiente ejemplo en BASIC es clarificador: PRINT 525 * 1570 * 270.

La memoria a corto plazo podría equivaler a los programas que tecleamos o introducimos en la memoria del computador procedentes de un cassette o un disquete.

Este tipo de información, se mantiene en memoria mientras no se desconecte la máquina; siempre, claro está, que deliberadamente no instruyamos al computador en otro sentido.



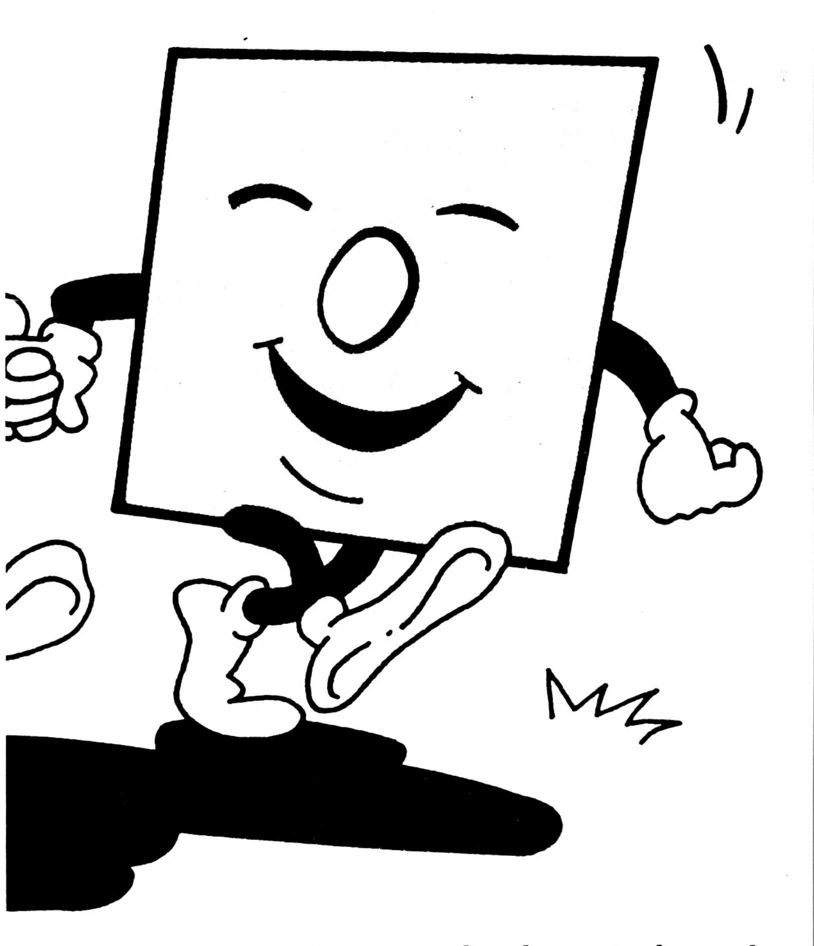
La información, sea cual fuere, se almacena dentro de la memoria en unidades del tipo *byte*, quedando a disposición del microprocesador.

A cada uno de los lugares donde se puede depositar un byte se le conoce como posición de memoria. Todas las **posiciones de memoria** está numeradas de tal forma que podemos referirnos a cualquiera de ellas en función de su dirección de memoria.

**Las direcciones de memoria** comienzan en la dirección 0 y pueden llegar —para un microprocesador de 8 bits— hasta la 65.535.

Un símil válido podría ser el de una calle con un vecino por casa y todas las casas en la misma acera. En este caso, cada vecino sería un **byte** o **unidad de información**.

Los habitantes de cada casa están representados por diferentes combinaciones de **8 bits.** Es decir, por un **byte**.



Dentro del computador, la parte de su electrónica destinada a guardar esta información se denomina RAM —acrónimo de Random Access Memory— y, en lenguaje informático, es conocida como "volátil", puesto que todo su contenido desaparece al privarla de energía eléctrica.

Otra característica de la memoria RAM es que tanto sirve para guardar información en ella como para extraerla, según nos interese.

La memoria a largo plazo se distingue de las otras dos en que la información que contiene no desaparece al cortar el suministro eléctrico al computador.

Dentro de este tipo podemos establecer la siguiente clasificación:

Situada dentro de la memoria interna y situada fuera del propio computador.

La memoria ROM —acrónimo de Read Only Memory— pertenece al primer tipo, ocupando una zona de la memoria interna total.

En la memoria ROM no se puede guardar información. Sólo se puede extraer, con lo cual su contenido permanece invariable en el tiempo.

El conjunto de programas contenidos en la ROM permiten que ciertas operaciones sean posibles al conectar la máquina, como veremos en su momento.

Finalmente, y aún dentro de la memoria a largo plazo, tenemos los soportes de memoria externos al computador, cuya misión es almacenar información y mantenerla a disposición de éste.

Ejemplos típicos son las casetes y discos magnéticos.

Gracias a los soportes externos, tenemos la posibilidad de transferir el contenido de la memoria RAM a estos, antes de desconectar la máquina, y, de esta forma, no perder su información.

Los *bytes* contenidos en la RAM, son transferidos secuencialmente al soporte externo para su almacenamiento, y, en sentido contrario y de la misma forma, son recuperados por la memoria RAM al cargar una información procedente del soporte externo.

En este sentido, es usual llamar **escritura** al proceso de grabación de información en el soporte externo, y **lectura** al de recuperación de la misma por parte de la memoria RAM.

Con el objeto de simplificar y dejar este último punto suficientemente claro, diremos que la RAM **lee** en el disco o **escribe** en él, según carge o grabe información en/o de el disco.

Para concluir con este epígrafe, se hace imprescindible resaltar que, mientras un ser humano puede sobrevivir con poca o ninguna memoria, un computador quedaría descalifi-

cado como tal, siendo manifiesta su incapacidad operativa.

## *ESQUEMA TIPO DE MICROORDENADOR. FLUJOS DE INFORMACION*

Las flechas indican el sentido en que circula la información cuando llega el momento de su transferencia entre ordenador y periférico.

**Periférico** es el nombre que, de forma genérica, recibe cualquiera de las máquinas susceptibles de ser conectadas con el computador.

## *TRANSMISION*

Página atrás, definimos la información como una transmisión de datos.

De aquí, parece poder deducirse que el efecto de transmitir resulta en un traslado de información de un ente físico a otro.

Al ente que emite la información se le llama **emisor.**

Al que la recibe**, receptor.**

También, ha quedado establecido que la memoria almacena información, y si suponemos que tanto el emisor como el receptor en la transmisión tienen memoria, el traslado de información correspondiente no lleva implícito que el emisor pierda el contenido de su memoria en beneficio de la del receptor.

En otras palabras, después de una transmisión, el emisor y el receptor comparten el conocimiento transmitido.

Nóisimsmart ed oidem le raidutse a somesap. Bueno, no se asuste, lo que he querido decir es: Pasemos a estudiar el **medio** de transmisión.

El único problema que le ha surgido es que he utilizado, como medio de transmisión de ese mensaje, la escritura en español, pero con las letras de cada palabra colocadas en orden inverso y las palabras invertidas.

Dicho de otra forma, yo, el **emisor** de información, me he comunicado con usted el **receptor**, a través de un **medio**, la escritura, pero con un **código** no acordado.

El **soporte** de esta información es el papel, o lo que es igual está memorizada mediante la escritura en papel.

En este punto se exige hacer una recapitulación.

Para ello, podemos decir que, para que una transmisión sea eficaz, requiere de **un emisor, un receptor, un medio, un código** y **ds soportes** de memoria (uno para el emisor y otro para el receptor).

Con estos antecedentes, apliquemos los conceptos anteriores a la transmisión de información entre el computador y una unidad de disco —disk drive— conectado a él; este periférico tiene reservado un capítulo. Consiguientemente, el puesto de **receptor** y **emisor** lo tomarán, según las circunstancias bien el computador bien el disco.

El **medio** es el sistema de enumeración en base dos, en forma de **bytes**.

El **código** es el ASCII, explicado anteriormente.

El **soporte de memoria del computador** es la RAM, volátil ya que su contenido se pierde en el momento de la desconexión.

El **soporte de memoria de la unidad de disco** son los discos, de los que hablaremos en otro epígrafe.

## *FICHERO. CONCEPTO*

Un fichero es un grupo de datos relacionados entre sí.

Nada impide que un fichero contenga un solo dato.

Dentro del ordenador, y con respecto a él, se individualiza cada información merced al concepto de fichero.

A cada fichero —o información agrupada e individual— se le asigna un nombre, el cual le distinguirá de entre los demás.

La denominación de un fichero está sometida a ciertas reglas que veremos en su momento.

Baste de momento saber que un fichero puede contener cualquier información:

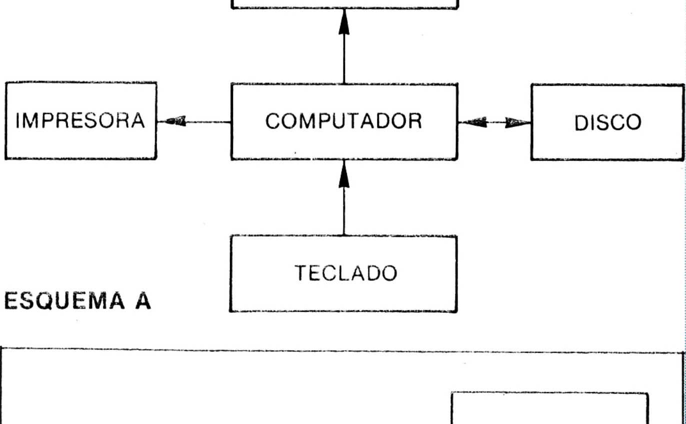
* Relación de los equipos de la liga de fútbol.
* Nombres de los participantes en un campeonato de mus.
* Un programa en BASIC.

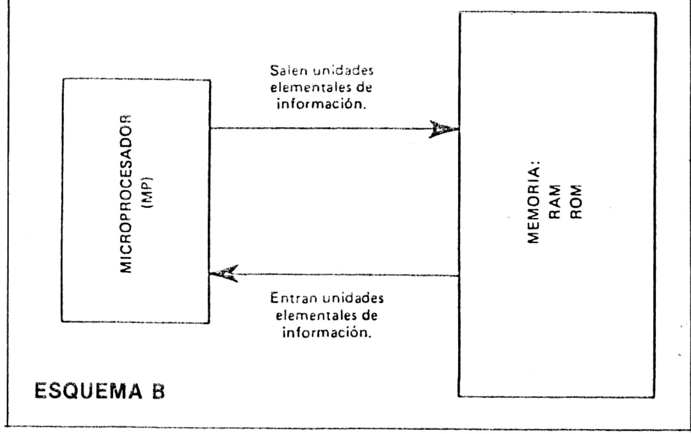
Y que nombres válidos para estos ficheros serían:

* LIGA
* MUS
* CUENCAS

Para concluir con esta breve introducción al concepto de fichero, añadiremos que toda transmisión de información entre la memoria interna del computador y un disco exige referirse a ella por su nombre —nombre del fichero— para diferenciarla de cualquier otra posible.

Estas ideas se irán consolidando y ampliando a lo largo de la lectura de los sucesivos epígrafes.




# Boletín de suscripción

A remitir a GTS. S. A. C/. Bailén, 20. 1.° Izda. 28005 Madrid

Deseo suscribirme a los 11 números anuales de Amstrad Educativo por sólo 2.500 pts.

El importe lo haré efectivo:

☐ **Por giro postal n.°** .......................

☐ **Por talón nominativo adjunto.**

☐ **Contra reembolso a la recepción del primer ejemplar, más gastos de envío.**

Deseo suscribirme a partir del n.° ....................... (inclusive).

Nombre y apellidos: ..........................................................

Domicilio: ..........................................................

Ciudad: .......................................... Teléfono ..........................

Fecha: .......................................... Firma ..........................

BASIC

# EL BASIC DEL AMSTRAD (II)

## CONCEPTOS PREVIOS

Un programa es una secuencia de instrucciones susceptibles de ser interpretadas por un computador, el cual las ejecutará, una tras otra, de principio a fin.

### Línea de programa

La secuencia de instrucciones que conforma el programa, se establece gracias a las líneas de programa, cada una de las cuales está estructurada de la siguiente forma:

N.° de línea Comando Argumento Fin de línea

El número de línea establece el orden de ejecución de las instrucciones, desde el menor al mayor, sin que por ello deban ser consecutivos.

Los números de líneas deben estar comprendidos entre 0 y 9999.

El comando puede ser cualquier palabra del BASIC, la cual, cumplirá su función sobre el argumento.

El *Fin de línea* se establece al apretar la tecla ENTER, con lo que la línea en cuestión pasa a la memoria del computador.

Una línea de programa puede estar formada por una sola sentencia como la esquematizada más arriba, o bien por varias, con lo que se tendría una línea multisentencia como la que sigue:

Número de línea Comando Argumento: Comando Argumento Fin de línea

Como puede apreciarse, una sentencia se separa de la siguiente por medio de dos puntos (:), excepto la última que requiere el ENTER para establecer el Fin de línea.

## Modos de operar

Cuando una sentencia es introducida en el AMSTRAD sin número de línea, se ejecutará inmediatamente al apretar ENTER. Esta forma de operar se conoce como MODO DIRECTO.

En modo directo también se pueden introducir multisentencias.

El MODO PROGRAMA exige introducir las sentecias con sus números de línea correspondientes y sólo serán ejecutadas secuencialmente mediante RUN y ENTER.

## Modo Programa. Cómo se introduce un programa

El objeto fundamental de este curso es mostrarte los fundamentos en que se basa la programación, pero, previamente, debes tomar contacto con algunos conceptos que, aun no entendiéndolos en su plenitud, han de comenzar a ser conocidos para evitar dudas.

Una vez diseñado un programa, y con el fin de introducirlo en el computador por primera vez, es necesario apretar todas y cada una de las teclas correspondientes a los caracteres escritos en las líneas del programa. Es decir, hay que teclear todos los números, letras, símbolos y signos escritos por el programador en cada línea del programa, incluidos los espacios.

Las mayúsculas y los caracteres situados en la parte superior de las teclas se consiguen apretando simultáneamente SHIFT y la tecla apropiada.

Un ejemplo te ayudará a comprenderlo mejor. Vamos a teclear una línea del programa:

30 INPUT "Longitud de la base: " ; b

Apretando cada una de las letras que contienen estos caracteres, hay que seguir los siguientes pasos:

— Teclear la palabra de la orden (INPUT) y otro espacio.

— Teclear la palabra del comando (INPUT) y otro espacio (opcional).

— Teclear las comillas ("), el texto ("Longitud de la base"), los dos puntos (:), un espacio, las comillas ("), punto y coma (;) y el nombre de la variable (b).

— Finalizada la línea, hay que apretar la tecla ENTER, RETURN o la que el MANUAL indique para que el computador la almacene en su memoria.

Al ir tecleando la línea, ésta va apareciendo en la pantalla del monitor.

Cuando, —una vez finalizada— se introduce en la memoria del computador, el cursor se queda sólo al comienzo de una línea en blanco.

Recuerda que, aunque las líneas del programa se tecleen en desorden, el computador siempre las memoriza siguiendo la secuencia de sus números.

Por último, debes saber cómo manejar algunos signos (el punto, la coma, los dos puntos, y el punto y coma) que, además de su valor ortográfico, tienen un valor diferente en el lenguaje de programación BASIC.

Todos estos signos tendrán valor ortográfico únicamente si son tecleados entre comillas (") dentro de una expresión alfanumérica.

En programación BASIC, estos signos cumplen diferentes funciones:

**El punto** (.) se utiliza para separar la parte entera de la parte decimal de un número, en lugar de la coma decimal. Por ejemplo, hay que teclear 2.53 en lugar de 2'53 o de 2,53.

**La coma** (,) se utiliza para:

— Separar los datos de salida en las sentencias PRINT. Por ejemplo: **PRINT A, B, C.** Teclea esta línea en modo directo para ver qué pasa.

— Separar los datos de salida en la instrucción DATA. Por ejemplo: **DATA A, B, C.** Teclea la línea en modo directo.

— Para separar dos expresiones obligando a que la segunda se imprima a partir de la siguiente zona de tabulación de la pantalla ocupada por la primera expresión. Por ejemplo: **PRINT "2+2=",2+2.** Tecleala en modo directo.

**Los dos puntos** (:) sirven para separar comandos en las líneas multisentencia. Por ejemplo: **INPUT A: PRINT A.** Tecleala como en los ejemplos anteriores.

**El punto y coma** (;) puede ser utilizado:

— Para separar datos, como las comas.

— Para obligar a que dos expresiones se impriman en pantalla una a continuación de la otra. Por ejemplo:

**10 PRINT "Me llamo";**
**20 PRINT "Juan"**

Al ejecutar este programa, tecleando **RUN** y **ENTER**, aparecerá impreso en pantalla: Me llamo Juan.

## Cómo se ejecuta un programa

Cuando hayas escrito un programa para resolver un problema y lo hayas tecleado, habrás puesto una secuencia de órdenes en la memoria del computador. Pero, para hacer trabajar este programa, es decir, para hacer que el ordenador obedezca una tras otra las instrucciones de la secuencia necesitas **correrlo** o **ejecutarlo**.

Para **correr** o **ejecutar** los programas, el BASIC tiene el comando **RUN** y, si a continuación, la tecla **ENTER**, harás comenzar la ejecución del programa que está en la memoria del computador, y podrás trabajar con él para resolver el problema planteado.

En caso de que al teclear una línea de programa cometas un error de sintaxis, el ordenador lo detectará y le avisará al tratar de ejecutar el programa, situándose el cursor en la línea que contenga el error. Entonces, deberás corregirlo siguiendo las instrucciones que te indicamos a continuación.

# DEPURANDO UN PROGRAMA

## Cómo se depura un programa

Es muy posible que cualquier programa que hayas escrito, tecleado y ejecutado por primera vez no funcione correctamente. Esto les suele suceder también a los mejores programadores y teclistas.

La causa de que el programa no funcione correctamente será —casi con seguridad— algún tipo de error que h. yas cometido al escribirlo o teclearlo. El propio ordenador te indicará el error cometido mediante un mensaje que aparecerá en pantalla. No te desanimes: dispones de medios para hacerlo funcionar correctamente.

La tarea de hacer funcionar correctamente un programa, aprovechando razonablemente la memoria del computador y corrigiendo su errores, se llama **depurar el programa**. Este trabajo podrás realizarlo utilizando las facilidades de **edición** que te ofrece el AMSTRAD.

Como primera norma, debes acostumbrarte a inicializar todo el sistema cuando vayas a comenzar a trabajar sobre un nuevo programa en otras palabras, limpiar la memoria interna del AMSTRAD de cualquier contenido previo. Esto lo conseguirás apretando las teclas CONTROL, SHIFT y ESC (CONTROL, MAY y ESC en el modelo con teclado español) simultáneamente. Otro sistema, más expeditivo y menos recomendable, es cambiar a OFF el interruptor que está marcado POWER a la derecha o detrás de la carcasa.

Hecho esto, y una vez comenzada la introducción de un programa, puede darse el caso de que una determinada línea no nos interese mantenerla en el listado y, consiguientemente, deseemos borrarla, para lo cual bastará escribir su número seguido de RETURN.

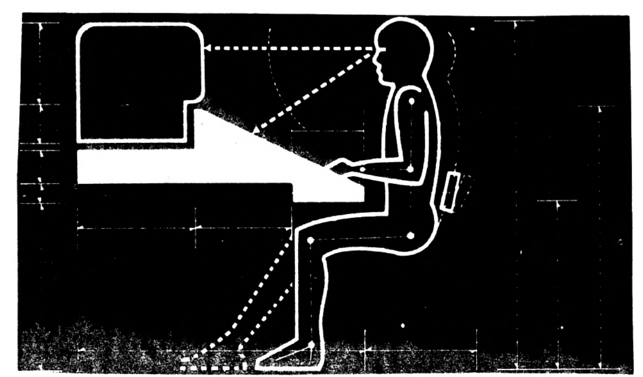
Para estudiar este caso, y otros que se propondrán más tarde, veamos un sencillo programa, el cual deberás teclear en tu computador exactamente como se indica.

Primero escribe

```
10 PRINT "En un lugarr de la Mancha"
20 STOP
30 PRRINT "de cuyo nombre quiero
   acordarme"
```

Al acabar cada una de estas líneas aprieta RETURN, con el fin de hacerla pasar a memoria y colocar el cursor al comienzo de otra línea.

Una vez hecho esto, escribe 20 seguido de ENTER; en este momento la línea 20 habrá desaparecido... aunque, de momento, no se pueda apreciar.



# Fichas del AMSTRAD (II)

POR A. BELLIDO

## *Guía de palabras BASIC*

*DEFREAL*

Vale todo lo dicho para DEFINT, pero referido a los números reales.

*DEFSTR*

Vale todo lo dicho para DEFINT, pero referido a las variables alfanuméricas.
DEFSTR B : b="ABCD" : PRINT B
ABCD

*DEG*

Ordena cambiar de radianes a grados.
Radianes es la unidad para iniciar el sistema.

*DELETE*
*DELETE —n.° de línea*
*DELETE n.° de línea—*

Borra todas, o desde el principio al **—n.° de línea**, o desde el **n.° de línea—** al final, las líneas de programas.

```
  10 DEFFN A(x,y)=x+y*2
  20 FOR x=1 TO 10 : LET Z=FN (A,10) :
PRINT Z;
  30 NEXT
  40  DELETE —10
```

Al ejecutar el listado, desaparece del mismo la línea 10.

*DERR*

Esta función guarda el número del código del último error derivado del disco.

*PRINT DERR*
*TIPO DE ERROR*

142 El rumbo (**stream**) no está adecuado.
143 ............ Fin de fichero en el disco.
144 .................. Comando incorrecto.
145 .................... Fichero existente.
146 ................. Fichero no existente.
147 ...................... Directorio lleno.
148 ......................... Disco lleno.
149 ...................... Disco cambiado.
150 .............. Fichero de sólo lectura.
154 ........ Fín de fichero (**End of file**).

*DI*

Impide cualquier tipo de interrupción en la ejecución de una parte de un programa.
Con **EI** se permite nuevamente las interrupciones.

*DIM ¡nombre (elementos)!*

Dimensiona una matriz denominada **nombre** reservando tantas variables como indica el número dado por **elementos**. Si no se especifica, se reservan 10.
El subíndice de la primera variable es 0.
DIM MULTI$ (2,3)
Reserva 12 variables denominadas MULTI$ (0,0) , MULTI$ (0,1),..., MULTI$ (2,3)

*DRAW ¡expresión 1, expresión 2! |, expresión 3||, expresión 4|*

Dibuja una recta desde la posición que a la sazón ocupe el cursor de gráficos y hasta la

posición determinada por **expresión 1** (abscisa) y **expresión 2** (ordenada).

Las abcisas (eje horizontal) van de 0 a 639: 640 **pixels**.

Las ordenadas (eje vertical) van de 0 a 399: 400 **pixels**.

La **expresión 3**,si se usa, determina el color de la tinta en el campo de 0 a 15.

Con la **expresión 4,** si se usa, se fija el modo lógico de impresión de **pixels** entre los que están en la pantalla y los que se activan con DRAW: Normal=0, XOR=1, AND=2, OR=3.

```
5 CLEAR: MODE 1
10 INPUT "tinta"; x: INPUT "modo"; z
20 DRAW 400,0,x,z: PLOT 439,0: DRAW 639,0
30 DRAW 0,0,x,z
40 IF INKEY$=" " THEN 35
50 GO TO 5
```

*DRAWR ¡expresión 1, espresión 2! | , expresión 3|| ,expresión 4 |*

Donde **expresión 1** y **expresión 2** sn desplazamientos relativos de abscisa y ordenada, y el resto coincide con lo dicho en DRAW.

En el ejemplo anterior cambiar esta línea

```
30 DRAWR 0,100,x,z
```

*EDIT ¡número!*

Muestra en pantalla la línea de programa cuyo número es **número** y queda en condiciones de ser manipulada.

Ver APENDICE 2.

*END*

Da por terminada la ejecución de un programa.

*EOF*

Detecta si el fin de un fichero, (End of file), en disco ha sido alcanzado.

*ERASE ¡matriz 1, matriz 2,...matriz n!*

Borra las matrices denominadas **matriz 1, matriz 2,...,matriz n,** dejando disponible la memoria equivalente.

*ERL*

Esta función guarda el número de línea donde se produjo el último error.

```
10 ON ERROR GOTO 30
20 q+a
30 PRINT ERL
40 STOP
```

*ERR*

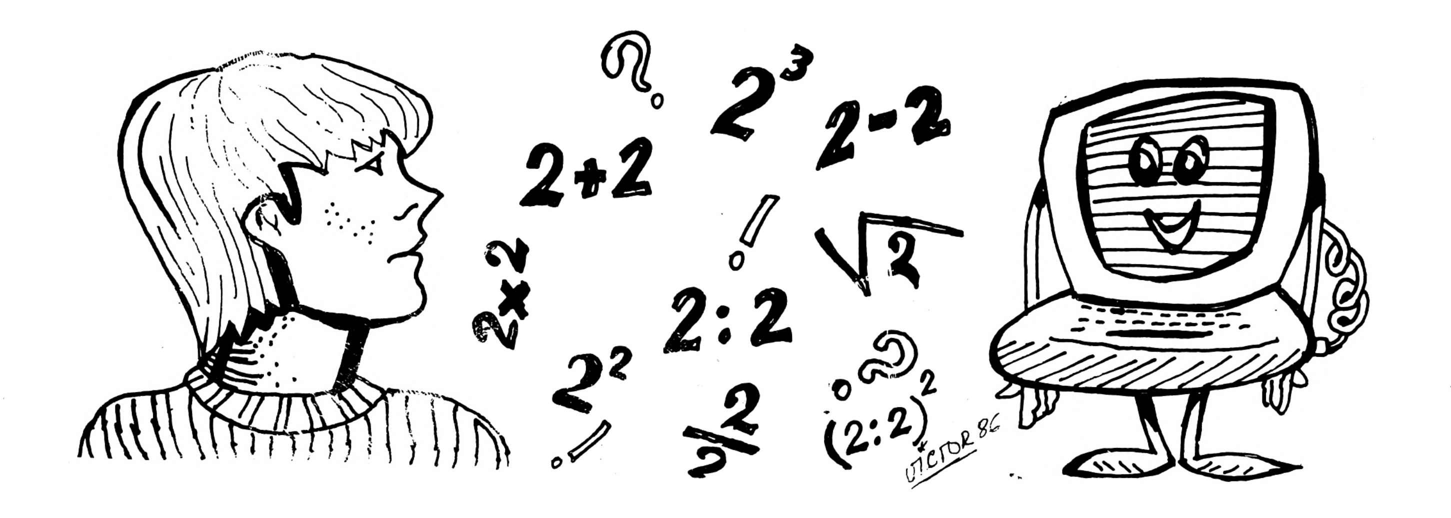
Esta función guarda el número del código del último error producido.

```
10 ON ERROR GOTO 30
20 PRINT ERL, ERR
40 STOP
```

*ERROR ¡expresión!*

Este comando genera, a todos los efectos, el tipo de error cuyo código es el dado por **expresión.**

```
5x=2
10 IF INKEY$ =" " THEN 10 ELSE ERROR x
```



# Programas comentados en

# BASIC

## *El juego de las 21 cerillas*

ESTE sencillo juego es ya un clásico, dentro de los que podemos denominar matemáticos o de base matemática.

A pesar de su sencillez, tiene la particularidad de ofrecer una estrategia ganadora, para el jugador que actua en segundo lugar.

### *REGLAS DEL JUEGO*

De un montón formado por 21 cerillas, los jugadores, por turno, deben tomar 1, 2, 3 ó 4 cerillas, de tal manera que pierde el que coje la última.

### *ESTRATEGIA*

Un jugador nunca pierde, si conoce la estrategia del juego, que se deriva del análisis de las propias reglas.

Observemos los números que intervienen y busquemos alguna relación entre ellos.

— Son 21 cerillas, recogidas 20 sobra 1 para el perdedor.

— Sólo pueden tomarse 1, 2, 3 ó 4 en cada turno.

Si realizamos las combinaciones de estos cuatro números tomándolos por parejas y sumándolos, obtendremos:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1+1 | 2 | 2+1 | 3 | 3+1 | 4 | 4+1 | **5** |
| 1+2 | 3 | 2+2 | 4 | 3+2 | **5** | 4+2 | 6 |
| 1+3 | 4 | 2+3 | **5** | 3+3 | 6 | 4+3 | 7 |
| 1+4 | **5** | 2+4 | 6 | 3+4 | 7 | 4+4 | 8 |

Y estableciendo la frecuencia con que se presenta, cada suma, veremos que:

| SUMA | N.° VECES QUE APARECE |
|---|---|
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| 5 | 4 |
| 6 | 3 |
| 7 | 2 |
| 8 | 1 |

La suma 5, es la más frecuente y aparece en todas las columnas de posibles combinaciones. ¿Qué nos indica esto? Pues dos cosas, primero que sea cual fuere el número de cerillas que tome el primer jugador, cojiendo el complemento a 5 (5 - elegidas por el primer jugador), puede siempre realizarlo y segundo, si mantiene esta forma de juego (estrategia) en la cuarta jugada se habrán retirado $5 \times 4 = 20$ cerillas, de manera que al primer jugador le quedará únicamente una, lo cual le hará perder.

Conclusiones:

De este razonamiento se desprenden tres cosas fundamentales:

1.ª El ganador debe comenzar a jugar en segundo lugar.

2.ª Cojerá siempre 5 —las elegidas por el primero.

3.ª El primer jugador pierde siempre, si se cumplen las dos primeras condiciones.

## *PROGRAMA*

Una vez conocida la estrategia, vamos a realizar un pgm. de manera, que el ordenador asuma el papel de segundo jugador. El primer jugador será sin duda un amigo, al que deseas gastar una pequeña broma, pues evidentemente se enfrenta a un rival invencible.

*Variables utilizadas*

C — Contador de juegos.

N — Número de cerillas que quedan en el montón.

I — Cerillas que toma el jugador.

X — Cerillas que toma el ordenador.

XS — Var. del INPUT para nuevo juego.

ZS — "VEZ" o "VACES" según que el número de partidas sea uno o más.

*Comentarios*

| | |
|---|---|
| 20 | Inicializa el contador de juegos. |
| 30 | Pantalla en 40 columnas. |
| 40-60 | Presentación. Reglas del juego. |
| 70-80 | Cabecera del juego. |
| 90 | Inicializa la var. N a 21. |
| 100 | Selección del número de cerillas a tomar por el jugador. |
| 110-120 | Control del INPUT. Recuerda que CHRS(7) provoca un pitido (BEEP) y CHRS(11), sube el cursor a la línea anterior, de esta forma borra el texto del INPUT. |
| 130 | Decrementa el montón en I unidades tomadas. |
| 140-150 | Escribe jugada del jugador y resto del montón. |
| 160 | Juega el ordenador, según estrategia descrita, y dec ementa el montón. |
| 170 | Escribe jugada realizada por el ordenador. |
| 180 | Controla el n.° de cerillas del montón. Si hay más de una el juego continua por la línea 100. |
| 190-210 | El juego ha terminado. Actualiza el contador C y ZS toma el valor apropiado (singular-1 juego o plural-varios). |
| 220-230 | Escribe el número de partidas perdidas (ya que ganadas no puede haber ninguna). |
| 240-250 | INPUT para nuevo juego y control del mismo. |
| 260-275 | Si no se va a jugar más, se poduce un comentario jocoso, en letras grandes (MODE ∅). CHRS(10) lleva el puntero abajo. |
| 280 | FIN DEL PROGRAMA. |
| 500-530 | Subrutina de continuación del juego. Los caracteres 224 y 225 corresponden a la cara seria y sonriente y pertenecen a los caracteres standard del Amstrad. |

## *MEJORAS*

Como todo pgm., es susceptible de mejoras y el expuesto anteriormente es muy elemental, te proponemos mejorarlo a fin de dar más cuerpo y vistosidad al propio programa y más desarrollo al juego.

— La primera mejora que se nos ocurre es obvia, utilizar color y gráficos ilustrativos. Un montón de cerillas representando en la pantalla del que van desapareciendo, las que se van tomando del mismo, le dará mayor realismo, puede añadirse además dos cajas o montones donde se depositen las tomadas por cada jugador.

— A fin de personalizar el juego, podía preguntar el pgm. el nombre del contrincante, de manera, que luego aparezca en las frases más o menos jocosas, del final de partida.

— La tercera, un poco más seria bajo el punto de vista de la programación, sería el dotar al ordenador de dos estrategias, la primera "ganadora" que es la que hemos desarrollado y la segunda, de tipo "aleatorio", es decir, que facilite el poderle ganar.

De esta forma, una vez tu amigo, se haya dado por vencido, podais vosotros jugar y ganar, para demostrar que no es tan difícil. Evidentemente habreis de recurrir a un truco sutil, que permita pasar al ordenador, de una estrategia a otra.



```
10 REM JUEGO DE LAS 21 CERILLAS
20 C=O
30 MODE 1
40 PRINT "DE UN MONTON CON 21
CERILLAS, PUEDES TOMAR CADA VEZ
1, 2, 3 ó 4"
50 PRINT :PRINT "EL QUE COJE LA
ULTIMA CERILLA ES EL PERDEDOR"
60 PRINT "PtPtPtPtPtPtPtPtPtPtPtPtPtPt
PtPtPtPtPtPtPtPtPtPtPtPtPtPtPt"
70 PRINT : PRINT " TU YO QUE DAN"
80 PRINT " ______________________
"
90 N=21
100 INPUT "Cuantas cojes";I
110 I =INT(I)
120 IF I<1 OR I>4 THEN PRINT CHR$(7)
+CH R$(11)+ "NO HAGAS TRAMPAS "
:GOTO 100
130 N=N-I
140 PRINT CHR$(11);
150 PRINT SPC(3); I;SPC(11);N
160 X=5-I : N=N-X
170 PRINT SPC(9); X;SPC(5);N
180 IF N>1 THEN GOTO 100
190 C=C+1
200 IF C=1 THEN Z$="VEZ " : GOTO 220
210 Z$ = " VECES"
220 PRINT :PRINT "HAS PERDIDO ";C;Z$
230 PRINT "=======================
================="
240 PRINT :INPUT "Quieres jugar denuevo
(S/N) ";x$
250 IF X$="S" OR X$="s" THEN GOSUB
500
260 MODE 0
265 PRINT :PRINT "NO HAS GANADO":
PRINT CHR$(10)+ "NI UNA PARTIDA!
"CHR$(225)
270 PRINT CHR$(10)+ "HACES BIEN DE-
JANDOLO" : PRINT CHR$(10)+ "Y DEDI-
CANDOTE A ...":PRINT CHR$(10)+"OTRA
COSA"
275 FOR K=1 TO 6:PRINT CHR$(10):NEXT
K
280 END
500 MODE 0:PRINT "Q U E M O R A L !"
510 FOR K=1 TO 300:LOCATE 10,9:PRINT
CH
CH R$(224):NEXT K
520 CLS: MODE 1: GOTO 70
530 RETURN
```

Envíanos tu programa mejorado, con estas u otras ideas y lo publicaremos.

# INTRODUCCION A LOS

## *FICHA NUMERO 1: CARGA DEL LENGUAJE LOGO EN EL AMSTRAD CP664*

En las fichas numeradas, que seguirán a algunos capítulos veremos gran variedad de temas. En principio, la única relación que unos de estos temas guardarán con otros, es que todos se referirán al lenguaje Logo.

Es una sección pensada para comentar todo tipo de cosas de interés. Aquí aparecerán análisis de programas, curiosidades, paradojas, experiencias, etc.

En general, decir que se va a cargar un programa en un ordenador, significa que se va a instalar ese programa en su memoria para poder trabajar con él. Ese programa podría ser un lenguaje capaz de generar otros programas. Así que, cargar el lenguaje Logo, significa instalar el lenguaje en la memoria del AMSTRAD, de modo que "comprenda" las palabras y la sintaxis de este lenguaje; después de lo cual podremos escribir y ejecutar programas en Logo.

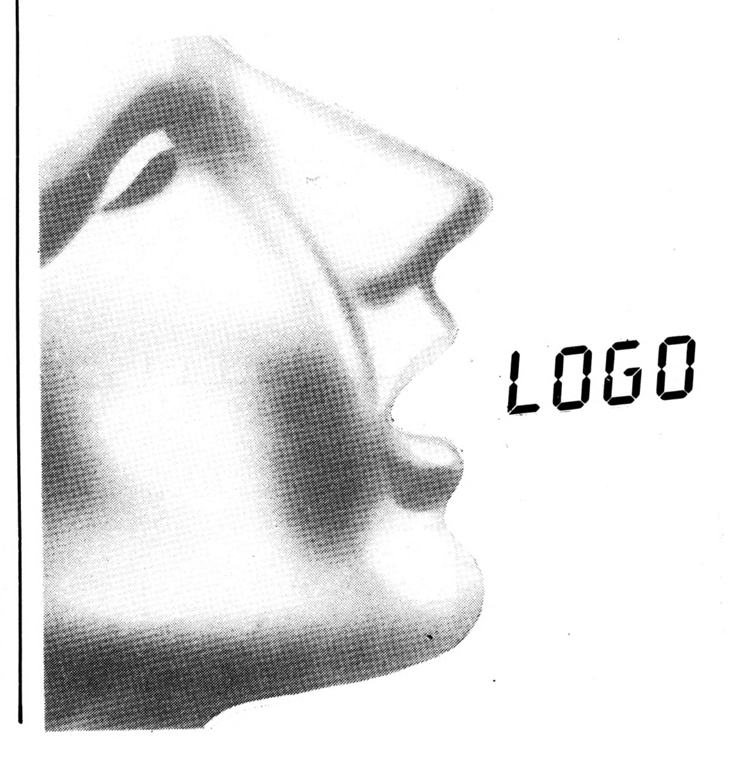
Con el ordenador se suministra un disco muy importante. Debemos poner mucha atención para que no se estropee. En la cara "A" contiene el sistema operativo CP/M y en la "B" el lenguaje DR. Logo.

Lo mejor es hacer copias de los discos importantes que poseamos. De este modo no se puede estropear el original, con las graves consecuencias que esto traería consigo.

Lo primero que vamos a ver en consecuencia, es la forma en que debemos proceder para copiar el disco.

No es este el lugar indicado para estudiar el concepto de sistema operativo. Bastenos saber, de momento, que para que un ordenador pueda ejecutar programas en distintos lenguajes (por ejemplo), es necesario que un programa "subyacente" gobierne algunas acciones de la máquina; por ejemplo, leer la información de los discos o escribirla en ellos.

Siempre que encendamos o apaguemos el AMSTRAD, debemos asegurarnos de que no hay disco alguno introducido en la unidad (podría resultar dañado). Al encender la máquina, se encuentra siempre bajo el control del sistema operativo AMSDOS, propio del AMSTRAD. Después de encender introduzcamos el disco suministrado, con la cara "A" hacia arriba. Para cargar el sistema operativo CP/M basta escribir |CP/M (el carácter | se obtiene pulsando simultáneamente las teclas SHIFT-@). Es indiferente que se escriba en mayúsculas o minúsculas. Después de algunos segundos aparece el mensaje.



# PROGRAMAS EN LOGO

POR JUAN MARTINEZ PINTOR

"CP/M 2.2-Amstrad Consumer Electronics plc", y el prompt del CP/M " A≦", con el cursor esperando órdenes.

El sistema operativo CP/M contiene, grabados en el disco, varios programas con utilidades diversas. Una de ellas es un programa que nos permite copiar el contenido de un disco en otro. Trabajando con una unidad de disco este programa es el DISCCOPY.

Para ejecutarlo, basta escribir en la consola el nombre del programa. Escribamos pues DISCCOPY, pulsemos la tecla "enter", y esperemos acontecimientos. A partir de este momento, la máquina está bajo el control del programa DISCCOPY, del sistema operativo.

Para ejecutarlo, basta escribir en la consola el nombre del programa. Escribamos pues DISCCOPY, pulsemos la tecla "enter", y esperemos acontecimientos. A partir de este momento, la máquina está bajo el control del programa DISCCOPY, del sistema operativo.

En todo el proceso de copia, cada vez que el ordenador se refiera al "SOURCE DISC" introducimos la cara "B" del disco suministrado con el equipo; cada vez que se refiera al "DESTINATION DISC", introducimos nuestro disco en blanco, en el que queremos copiar el anterior. El orificio protector de escrituras accidentales, debe estar abierto en el disco principal (para no permitir la escritura en él), y cerrado en el disco copia (para poder escribir en él).

La operación de sacar un disco y meter otro, ha de repetirse varias veces, porque cada vez se copian 8 pistas y son 40. Cuando se termine el proceso de copia, el programa preguntará si se va a hacer alguna otra copia. Si contestamos que no, el ordenador pasará de nuevo a estar bajo el control del sistema operativo.

Vamos a ver a continuación como poner al ordenador en situación de trabajar con Logo.

Se puede cargar el lenguaje Logo, tanto si la máquina está bajo el sistema operativo CP/M como bajo el AMSDOS (al encenderla por ejemplo).

Bajo AMSDOS, una vez introducido el disco que contiene el Logo, basta escribir |CP/M (como hicimos antes para cargar el CP/M, con la otra cara). Bajo el CP/M, basta escribir la palabra LOGO.

En ambos casos el ordenador lee en el disco el programa que contiene el lenguaje Logo, y después de unos segundos, lo ejecuta. Esto equivale a que el control del aparato pasa a tenerlo el lenguaje Logo. Aparecerá entonces la pantalla borrada y una interrogación "?". Es el "prompt" del Logo. A su derecha el cursor (un cuadrado) esperando nuestras órdenes para empezar a trabajar.

## *CAPITULO II*
## *MODOS DE OPERACION*

* Una "instrucción" es una palabra o lista de palabras que indica al ordenador lo que debe hacer.

* Las instrucciones que provocan una acción, se llaman "órdenes".

* Las instrucciones que producen un resultado, se llaman "operaciones".

* Toda instrucción del tipo operación ha de ir precedida por una del tipo orden que la afecte, o por otra operación.

* Los "modos de operación" son los estados distintos en que puede estar el ordenador al trabajar en Logo.

* Cuando se carga Logo en el AMSTRAD, se sitúa en un modo desde el que se accede a todos los demás. Se llama "nivel superior" o "toplevel".

* Al comenzar la ejecución de un programa, el Logo abandona el nivel superior. Es el "modo ejecución".

* Un "procedimiento" es un programa escrito en Logo.

* Otro modo de operación es el "modo definición". Se usa para definir los nuevos procedimientos.

* Todo procedimiento consta de tres partes: "nombre", "cuerpo" y "final".

* El nombre es una palabra: identifica al procedimiento a todos los efectos.

* El cuerpo consiste en todas las instrucciones que forman el procedimiento.

* El final indica donde termina el procedimiento.

* Para entrar en modo definición, se escribe la palabra "to" seguida del nombre del nuevo procedimiento.

* Para concluir la definición se escribe la palabra "end" al principio de una nueva línea.

* Si se pulsa la tecla "esc" el Logo abandona lo que esté haciendo.

* El "modo edición", se puede utilizar para definir procedimientos y para modificar los ya definidos.

## *II.1. TIPOS DE INSTRUCCIONES: ORDENES Y OPERACIONES*

Según se ha visto en el capítulo 1, al comenzar a trabajar en Logo el ordenador conoce unas cuantas docenas de palabras.

Una instrucción es una palabra o lista de palabras, que indica al ordenador lo que debe hacer cuando la ejecute. El Logo cuenta entre sus características el hecho de que es posible definir nuevas palabras y, por tanto, nuevas instrucciones, en función de otras que ya conoce o que se definirán posteriormente.

Cuando el ordenador ejecuta un programa escrito en un determinado lenguaje, realiza acciones de muy distinta naturaleza. Por ejemplo, borrar la pantalla, realizar cálculos,



escribir resultados, hacer un gráfico. Estos ejemplos son suficientes para percibir que podemos diferenciar las instrucciones que producen un resultado, de aquellas otras que no lo producen.

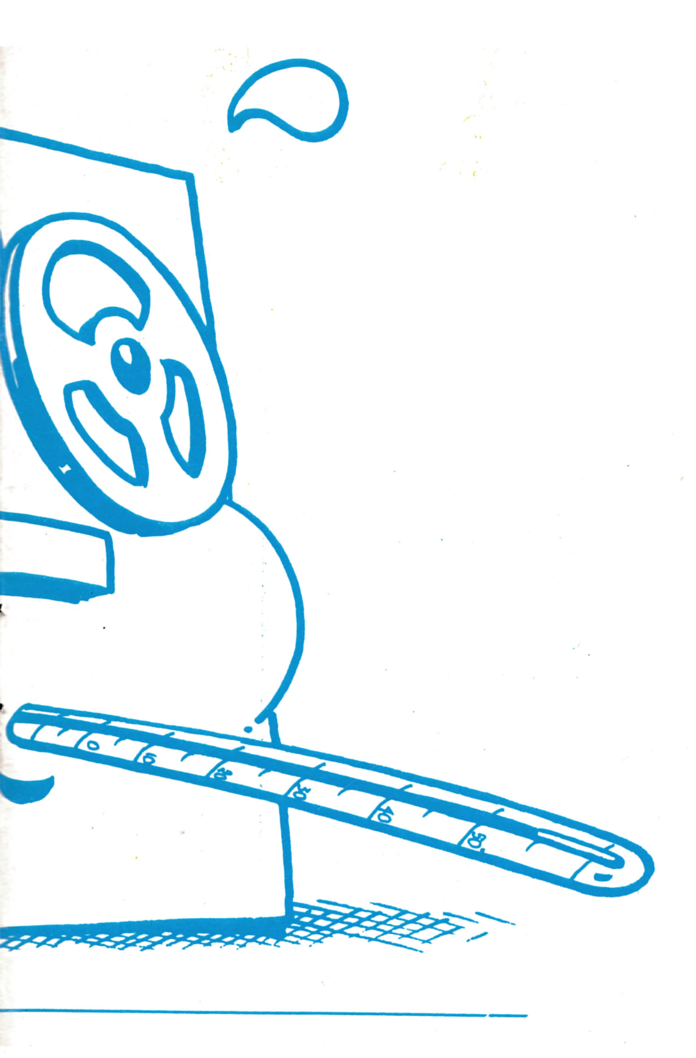
Las instrucciones que provocan que el ordenador lleve a efecto una acción, se llaman **ORDENES**. Por el contrario, las instrucciones que al ser ejecutadas, producen un resultado, se llaman **OPERACIONES**. Unas y otras son o bien primitivos o bien procedimientos. Es decir, el programador de Logo puede definir nuevas órdenes y nuevas operaciones.

**EJEMPLO II.1**

— Una instrucción que haga que la pantalla se borre es una orden. No produce resultado alguno; produce una acción.

— La instrucción que se utiliza para sumar dos números es una operación; da como resultado la suma de estos dos números.

Podemos utilizar y entremezclar en nuestros programas ordenes y operaciones, teniendo siempre presente la diferencia



conceptual existente entre ambas. En efecto, hay un hecho que debemos mantener siempre presente:

"Toda instrucción del tipo operación ha de ir precedida por una orden que la afecte e indique lo que se debe hacer con el resultado producido, o bien por otra operación uno de cuyos datos sea el resultado producido".

De otra manera, se produciría un error o no se haría nada con el resultado de la operación.

**EJEMPLO II.2**

— Para la suma se utiliza el signo "+". Una instrucción del tipo operación podría ser entonces "3+98". Pero si se escribe en un programa una instrucción así, aunque el ordenador calcule correctamente el resultado, no sabrá que ha de hacer con él, puesto que no se le ha indicado, haciendo preceder la operación por una orden.

## *II.2 ORDENES DIRECTAS: "TOPLEVEL"*

Cuando el ordenador se prepara para trabajar en un lenguaje, en general, puede hallarse situado en uno de varios estados o modos, según lo que se este haciendo en cada momento. No es lo mismo, por ejemplo, que se este ejecutando un programa o que se este definiendo uno nuevo. Estos estados distintos, son lo que llamamos "modos de operación".

Cuando se carga el lenguaje Logo en el AMSTRAD, se situa este en un "modo" desde el que se puede acceder a todos los demás. Es el modo llamado **NIVEL SUPERIOR** o **TOPLEVEL**. A este estado es al que regresa la máquina cuando termina la ejecución de los programas. Y es a partir de este estado desde el que comienza su ejecución.

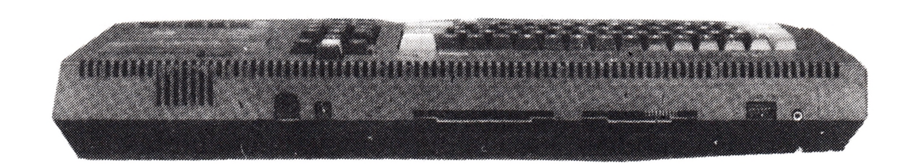
Además, cuando el ordenador se halla en toplevel permite al usuario la "ejecución directa" de muchas instrucciones. Para ejecutar una instrucción desde el nivel superior, solo tenemos que escribirla en el teclado, y después pulsar la tecla "enter". Entonces la instrucción se ejecuta y, a continuación, se regresa al nivel superior.

Podemos distinguir el estado en que esta la máquina, por algunas características de la pantalla. Así, cuando esta en toplevel, vemos que en la línea en la que se escribiran los próximos caracteres que se tecleen, aparece un signo de interrogación. Este carácter se llama **PROMPT**. A su derecha vemos un cuadrado, que nos indica donde se escribira el próximo carácter. Es el **CURSOR**.

En nivel superior podemos incluso escribir varias instrucciones seguidas, para que se ejecuten consecutivamente. Basta para ello, no pulsar enter entre ellas, y separarlas por un espacio en blanco. Puede suceder que se nos acabe la línea de escritura (el renglón), en un momento en que nos deje una palabra dividida en dos. Entonces el ordenador escribirá el caracter "!" al final de la línea y podremos seguir como si nada pasara. La palabra queda entonces dividida ante nuestra vista, pero no a la vista del ordenador, que la ejecutará normalmente. Esto sin embargo solo puede hacerse una vez antes de pulsar enter: es decir, si se siguen escribiendo instrucciones en el renglón de abajo, al acabarlo se tendrá que pulsar enter, porque el ordenador no permite seguir escribiendo. Esta limitación es válida no solo en el nivel superior, como ya veremos.

# DOCTOR LOGO

## Programas comentados en LOGO



### DISPARO (LOGO)
### (Movimiento Absoluto) (DD)

**Instrucciones de manejo:**

Enciéndase la máquina e introduzcase el disco que contiene el Dr. Logo. Escríbase ICMP y el lenguaje Logo se instalará en el ordenador.

Para cargar el programa basta escribir **load "disparos**. Después de unos segundos el programa se carga en la memoria. El ordenador informa en pantalla de los nombres de los procedimientos cargados:

diana defined
inicio defined
asigp defined
res defined
disparos defined

Como el lenguaje Logo consume mucha memoria no se debe cargar este programa después de haber estado trabajando con Logo en otra cosa. Lo mejor es reinicializar antes el sistema.

Para comenzar a jugar escríbase el nombre del programa: disparos. Después "enter". Aparece en la pantalla un resumen de las instrucciones. Cuando se haya leído basta pulsar una tecla para que el programa continue.

El juego consiste en realizar 10 disparos con la tortuga a una di na que aparece en pantalla. Las posiciones de la tortuga y la diana las elige el ordenador al azar en cada jugada.

Cuando el ordenador pide los datos del disparo se deben escribir dos números separados por un espacio en blanco y luego "enter". El primer número indica la inclinación que debe tomar la tortuga para dar en la diana. Cuando mira hacia arriba la tortuga tiene una inclinación de 0 grados. La orientación aumenta de 1 en 1 en el sentido de giro de las agujas del reloj. De modo que si se desea que mire exactamente hacia la derecha se debe dar una inclinación de 90 grados. El segundo número representa el número de pasos que se estima debe recorrer la torguga para dar en el blanco. Para hacer la estimación lo mejor es la experiencia; téngase, no obstante, en cuenta que la pantalla tiene 640 pasos de tortuga desde su extremo izquierdo al derecho.

Por ejemplo, si para dar en el blanco se estima que la tortuga debe tener una inclinación de 32 grados y que debe recorrer 208 pasos, entonces basta escribir, en la línea de datos:

32 208

y luego "enter". Se verá en pantalla el camino recorrido por la tortuga y la posición final (a menos que se salga de los límites de la pantalla). En la línea de datos aparecerá la puntuación obtenida en ese disparo. Pocos segundos después aparecerá también el número de puntos acumulados hasta ese momento del juego, y el número de jugadas.

En caso de dar en la diana se obtienen 35 puntos. Según la distancia sea mayor se van perdiendo puntos.

Al final de los 10 disparos se termina el juego. En pantalla aparece el número total de puntos y la puntuación obtenida de 0 a 10.

```
① to disparos
② setpal 0[2 0 1]
③ setpal 1[2 1 0]
④ inicio
⑤ label "regr
⑥ make "num :num+1
⑦ jugada
⑧ setsplit 1 ss
⑨ type [PUNTOS:] type :puntos
⑩ type "
⑪ type [JUGADAS:] type :num
  wait 250
⑫ if :num<10 [go "regr]
⑬ res
⑭ end
⑮ to inicio
  cs ct ts
⑯ repeat 15[type "   ]
  pr "DISPAROS
  repeat 15[type "   ]
  repeat 8[type "   ]
  repeat 4[pr []]
⑰ pr [* APARECERA UNA DIANA EN ALGUN
    LUGAR DE]
  type ". type "
  pr [LA PANTALLA]
  pr [* EN OTRO LUGAR APARECRA LA
    TORTUGA]
  type [* SE TRATA DE HACER DISPA
    ROS CON LA TOR]
  (type " [TUGA DE MODO QUE SE ACER
    QUE TODO LO]) pr []
  type ". type "
  pr [POSIBLE A LA DIANA]
  pr [* PARA ELLO BASTA ESCRIBIR CU
    ANDO SE PI]
  type ". type "
  pr [DA LA ORIENTACION QUE DEBE
    TOMAR Y LA ]
  type " type "
  pr [DISTANCIA QUE DEBE RECORRER]
  pr [* ESCRIBASE LA ORIENTACION (0
    A 359) Y]
  type " type "
  pr [LA DISTANCIA SEPARADOS POR UN
    ESPACIO]
  make "puntos 0
⑱ make "num 0
  repeat 3[pr []]
  type [PULSAR ENTER]
  make "se rc
⑲ end
  to jugada
  setpc 1
⑳ cs fs
  diana
  sit
  setsplit 1 ss
㉑ setpal 2[2 2 0]
  type [ESCRIBIR LOS DATOS DEL DIS
    PARO:]
  make "intento rl
㉒ setpc 2 fs
㉓ sound [1 20 50]
  seth first :intento
㉔ fd first bf :intento
㉕ repeat 3[ht wait 40 st wait 15]
㉖ make "posfi list first tf first
    bf tf
㉗ ss
  make "disx (- first :pos first
    :posfi)
㉘ make "disy (- first bf :pos first
    bf :posfi)
  make "dx :disx*:disx
㉙ make "dy :disy*:disy
  make "dc :dx+:dy
㉚ asigp
  end
  to diana
  ht
  make "dx random 320
㉛ make "dy random 200
  make "az1 random 2
  make "az2 random 2
  if :az1=1 [make "dx -:dx]
  if :az2=1 [make "dy -:dy]
  if :dy<-184 [make "dy :dy+16]
㉜ make "pos list :dx :dy
  dot :pos
㉝ pu setpos :pos
  seth 0
  bk 10 pd lt 90 fd 10 rt 90
  repeat 4[fd 20 rt 90]
㉞ end
  to sit
  ht
  make "tx random 320
  make "ty random 200
  make "rx random 360
  make "az3 random 2
  make "az4 random 2
  if :az3=1 [make "tx -:tx]
㉟ if :az4=1 [make "ty -:ty]
  if :ty<-184 [make "ty :ty+16]
  make "tos list :tx :ty
  pu setpos :tos pd
  seth :rx st
  end
  to asigp
  if :dc>10000 [make "inc 0]
㊱ if and (or :dc<10000 :dc=10000)
    :dc>8000 [make "inc 2]
㊲ if and (or :dc<8000 :dc=8000)
    :dc>6000 [make "inc 4]
  if and (or :dc<6000 :dc=6000)
    :dc>4000 [make "inc 7]
  if and (or :dc<4000 :dc=4000)
    :dc>2000 [make "inc 9]
  if and (or :dc<2000 :dc=2000)
    :dc>1000 [make "inc 11]
  if and (or :dc<1000 :dc=1000)
    :dc>500 [make "inc 15]
```

```
if and (or :dc<500 :dc=500)
  :dc>200 [make "inc 18]
if and (or :dc<200 :dc=200)
  :dc>50 [make "inc 20]
if and (or :dc<50 :dc=50)
  :dc>10 [make "inc 23]
if and (or :dc<10 :dc=10)
  :dc>1 [make "inc 24]
if :dc=1 [make "inc 25]
if :dc=0 [make "inc 35]
type [PUNTOS DEL DISPARO:]
type :inc
```

```
㊳ make "puntos :puntos+:inc
wait 250
ct ss
end
to res
ts ct
type [PUNTOS:] pr :puntos
㊴ pr [] pr []
make "punt int(100*(:puntos/35))
type [PUNTUACION (0 A 10):]
pr :punt/100
end
```

# COMENTARIOS

**NOTA:** Los números rodeados con un círculo se corresponden con los comentarios del final del programa.

① Es el procedimiento principal
② Pone color del fondo
③ Pone color a los graficos
④ Esta es la llamada al procedimiento inicio
⑤ Marca el lugar al que debe volver el control tras un "go"
⑥ Asigna a la variable "num" un nuevo valor
⑦ Llamada a procedimiento
⑧ Indica el numero de lineas de la pantalla mixta.Instala esta pantalla.
⑨ Escribe en pantalla sin retorno de carro
⑩ Escribe un espacio en blanco
⑪ Produce una pausa
⑫ Si el valor de la variable "num" es menor que 10 el control pasa al lugar marcado con "regr"
⑬ Llamada a procedimiento
⑭ Final del procedimiento
⑮ Borra los graficos.Borra el texto instala la pantalla de texto
⑯ Escribe 4 lineas en blanco
⑰ Instala el valor 0 en la variable "puntos"
⑱ Instala en la variable "se" el siguiente caracter que se escriba en el teclado
⑲ Selecciona la pluma 1 para los graficos
⑳ Ordena que la ventana mixta tenga una linea de texto.Selecciona la pantalla mixta
㉑ Almacena en la variable "intento" la lista de datos que se escriba a continuacion en el teclado
㉒ Emite un sonido
㉓ Hace mirar a la tortuga en la direccion absoluta dada por el primer elemento de la lista "intento"
㉔ Hace avanzar a la tortuga tantos pasos como indique el segundo elemento de la lista
㉕ La tortuga hace intermitencia
㉖ Almacena en la variable "posfi" las coordenadas de la tortuga tras el disparo
㉗ Resta el primer numero de la lista "posfi" del primer numero de la lista "pos" y guarda el resultado en la variable "disx"
㉘ Hace que "dx" a;macene el cuadrado de "disx"
㉙ La variable "dc" almacena el cuadrado de la distancia entre tortuga y diana
㉚ Almacena en la variable "dx" un numero elegido al azar y comprendido entre 0 y 319
㉛ Para que la linea de datos no oculte a la diana, si su coordenada y es menor que 184, se aumenta en 16
㉜ Dibuja un punto en la posicion cuyas coordenadas almacena la variable "pos"
㉝ Dibuja el cuadrado que rodea a la diana
㉞ Si la variable "az3" vale 1, se toma negativo el valor de "tx"
㉟ Si el cuadrado de la distancia ("dc"), es mayor que 10000, se otorgan 0 puntos al disparo
㊱ Si "dc" es menor o igual que 10000 y mayor que 8000 se otorgan 2 puntos
㊲ Escribe el valor de la variable "inc" que son los puntos del ultimo disparo
㊳ Escribe el total de puntos conseguidos

# *El programa técnico del mes*

# SISTEMAS DE UNIDADES

**Autor: Victor J. Campo López**

## GENERALIDADES

La utilización de unidades de medida en varios sistemas y su conversión de unos a otros, origina en muchos casos, errores en los cálculos, a menos que se domine bien, el manejo de los factores de conversión. Estos errores, suponen un pequeño martirio para estudiantes y profesionales, que tratan a menudo con diferentes unidades.

De todos son conocidos los formularios, agendas, prontuarios, etc., usados comúnmente para salvar esta dificultad.

Tres son los sistemas adoptados y reconocidos universalmente, el Sistema CEGESIMAL (C.G.S.), el GIORGI, también conocido como MKS, y por último el sistema TECNICO.

El programa que proponemos, tiene cuatro opciones:

1.— GRAFICO
2.— CALCULO
3.— EJERCICIOS
4.— PARAR

— La primera opción, ofrece un gráfico con todas las unidades de los tres sistemas, relacionadas con unas reglas nemotécnicas, fáciles de recordar por su extremada sencillez. A la vista del cuadro y de acuerdo con las instrucciones que se ofrecen, en forma de textos móviles, pueden realizarse toda clase de conversiones, con suma facilidad.

La finalidad última de dicho gráfico, es la de poderse memorizar, de esta manera en cualquier momento, podremos hacer operaciones de cambio, sin tener que recurrir a ningun manual.

— La segunda opción, de tipo práctico, ofrece la posibilidad de realizar las operaciones de conversión, con el ordenador. El programa le permitirá elegir que tipo de unidad desea, y que cantidad, calculando las correspondientes a los otros dos sistemas.

— La opción tercera, es un complemento del gráfico, en el sentido de que le permite practicar las conversiones, mediante unos ejemplos que el propio ordenador le facilitará, donde él mismo asumirá el papel de profesor, indicándole sus aciertos y fallos (admitirá un márgen de error de una centésima en sus respuestas), así como el número de ejercicios realizados.

— Por último, la cuarta opción, permite detener el programa, de manera que puede acceder al listado, si así lo desea, e incluso introducir variaciones en el mismo.

En las tres opciones básicas, se utilizarán unicamente unidades de Masa, Fuerza, Trabajo y Potencia. Las unidades de Longitud, Tiempo, etc., se consideran tan sumamente fáciles de manejar en los tres sistemas, que suponemos son del dominio de todos.

## DESCRIPCION DEL GRAFICO. OPCION 1

El gráfico propuesto se compone de tres triángulos, que representarán de menor a mayor, a:

1) Unidades de MASA
2) Unidades de FUERZA
3) Unidades de TRABAJO Y POTENCIA

Cada vértice de los triángulos, corresponde a un sistema de unidades. El superior es el C.G.S., el izquierdo el GIORGI o MKS y el derecho el TECNICO.

Las unidades correspondientes se sitúan junto a los vértices. En los lados, se han reflejado los factores de conversión entre unidades, que se utilizarán, de acuerdo con el sistema de signos, indicado en el propio gráfico (izquierda o arriba — multiplicar, derecha o abajo — dividir).

## GRAFICO

Los factores de conversión resultan muy fáciles de recordar. Observe que todas las bases de los triángulos, tiene como factor <9.8>, esto indica que para pasar del sistema GEORGI al TECNICO, habrá que dividir por 9.8 y para pasar del TECNICO al GIORGI, multiplicar por 9.8, con independencia de que las unidades sean de masa, fuerza, trabajo o potencia.

Por tanto, recuerde que el factro 9.8, es común a las bases de los tres triángulos.

En los lados izquierdos, existen tres factores, el primero 10↑3 para el interior (masa), 10↑5 para el medio (fuerza) y 10↑7 para el triángulo exterior.

Para recordar estos números, observe que en el triangulo interior, se encuentra el gr.m y el Kg.m, su relación es evidente y Kg=1000 gr, es decir 10↑3, y los otros tienen como factor, las siguientes potencias impares de 10 (5 y 7).

Conocida la relación anterior, la correspondiente a los lados derechos, resulta aún mas sencilla, estos factores son sencillamente el producto de los otros dos lados, en cada triángulo.

Como conclusión, podemos decir, que son solo dos los números a recordar: el 9.8, común a los tres ubicado en las bases y el $10^3$, que es la relación entre el kilogramo y el gramo, conocido por todos.

Estos dos números, el convenio de signos y por supuesto conocer a que sistema corresponde cada unidad, nos permitirá, de forma sencilla, obtener rápidamente la conversión entre unidades.

Quizá le sirva la ayuda, recordar que las unidades menores, c rresponden al sistema C.G.S. y la mayores al TECNICO.

La opción 1, del programa, le dará una explicación más completa del gráfico, con ejemplos.

SIGNOS
x
:
x
:
C.G.S.
Ergio Ergio/s
1.— MASA
2.— FUERZA
3.— TRABAJO Y POTENCIA
3
dina
2
$9,8 \times 10^7$
gr.m
$10^7$ $10^5$ $10^3$ 1
$9,8 \times 10^5$
$9,8 \times 10^3$
Kg.m
9,8
UTM
Nw
9,8
Kp
GIORGI O MKS
Julio
9,8
TECNICO
Jul/s = watio
Kgm
Kgm/s

```
10 ' ......................................
15 ' .                                    .
20 ' .        SISTEMAS DE UNIDADES        .
30 ' ......................................
32 '
33 CLS:GOSUB 8000: FOR K=1 TO 1000:NEXT K
35 DIM U$(3,3) :GOSUB 1500
36 DIM PRE$(20),R(20):GOSUB 2500
40 MODE 1:GOTO 9000
50 MODE 0
55 PRINT CHR$(24)
56 LOCATE 5,9: PRINT "          "
60  LOCATE 5,10: PRINT " F  I  N "
70 LOCATE 1,17:END
100 ' .........  GRAFICO ...........
102 INK 0,0 :INK 1,20:BORDER 1:CLS
105 MODE 2: RESTORE 160
110    FOR T=1 TO 3
120 READ X,Y:PLOT X,Y
130    FOR P=1 TO 3
140 READ X,Y: DRAW X,Y
150  NEXT P,T
160 DATA 152,104,485,104,320,368,152,104
170 DATA 208,136,429,136,320,304,208,136
180 DATA 280,184,357,184,320,240,280,184
190    FOR P=1 TO 29
200 READ X,Y,Z$
210  LOCATE X,Y: PRINT Z$
220  NEXT P
230  PRINT CHR$(24):LOCATE 38,1: PRINT " C.G.S. "
235  LOCATE 62,20: PRINT " TECNICO "
240 LOCATE 6,20: PRINT " GIORGI O MKS ": PRINT CHR$(24)
250 DATA 35,2,"Ergio   Ergio/s"
260 DATA 23,20,"Julio",16,21,"Jul/s=watio"
270 DATA 60,21,"Kgm   Kgm/s"
280 DATA 40,4,"3",40,8,"2",40,12,"1"
290 DATA 39,6,"dina",25,18,"Nw",53,18,"Kp"
300 DATA 39,18,"9.8",39,20,"9.8",32,15,"Kg.m   9.8   UTM"
310 DATA 39,10,"gr.m",27,11,"7",32,11,"5",37,11,"3"
320 DATA 25,12,"10",30,12,"10",35,12,"10"
330 DATA 50,9,"9.8*10^7",47,11,"9.8*10^5",44,13,"9.8*10^3"
340  LOCATE 2,4: PRINT "x ————————————"+CHR$(243)+" :"
350  LOCATE 2,10: PRINT "x "+ CHR$(242)+"———————————— :"
360   LOCATE 2,5: PRINT CHR$(240)+"                    "+CHR$(148)
370 LOCATE 2,6: PRINT CHR$(148)+"                    "+CHR$(148)
380 LOCATE 2,7: PRINT CHR$(148)+"                    "+CHR$(148)
390 LOCATE 2,8: PRINT CHR$(148)+"                    "+CHR$(148)
400 LOCATE 2,9: PRINT CHR$(148)+"                    "+CHR$(241)
410 LOCATE 5,2: PRINT "SIGNOS"
420 LOCATE 58,1: PRINT "1 - MASA"
430 LOCATE 58,3: PRINT "2 - FUERZA"
440 LOCATE 58,5: PRINT "3 - TRABAJO Y POTENCIA"
450   FOR P=1 TO 3500: NEXT P
480 GOSUB 9500
490 PRINT CHR$(24)
500 STOP
1000 ' ............. C A L C U L O .................
1010 MODE 1
1015 LOCATE 3,1: PRINT "SELECCIONE TIPO DE UNIDADES"
```

```
1020      FOR K=1 TO 3: LOCATE 10,K*3
1030 PRINT K;" - ";U$(K,0): NEXT K
1040 Z$=INKEY$:IF Z$="" THEN  GOTO 1040
1050 T1=VAL(Z$)
1060 IF T1<1 OR T1>3 THEN  GOTO 1040
1062 CLS: PRINT "    QUE UNIDAD DE ";CHR$(24);"   ";U$(T1,0);"   ";CHR$(24): PRINT
"DESEA CONVERTIR ?"
1063 PRINT : PRINT
1064      FOR K=1 TO 3: PRINT TAB(10);K;" ) ";U$(T1,K): PRINT : NEXT K
1070 Z$=INKEY$:IF Z$="" THEN  GOTO 1070
1080 T2=VAL(Z$)
1085 IF T2<1 OR T2>3 THEN  GOTO 1070
1090  PRINT "QUE CANTIDAD DE ";U$(T1,T2);
1095 INPUT C
1096  LOCATE 1,17
1100 ON T1 GOTO 1110,1200,1300
1110 IF T2=1 THEN  PRINT C; "gr.m=";c/1000;" Kgm = ";c/1000*9.8;" UTM"
1120 IF T2=2 THEN  PRINT C; " Kg.m=";1000*c ;" gr.m = ";c/9.8;" UTM"
1130 IF T2=3 THEN  PRINT C; " UTM = ";C*9.8;" Kg.m = ";c*9.8*1000;" gr.m"
1140  GOSUB 9980:GOTO 9000
1200 IF t2=1  THEN  PRINT C;" Dinas = ";c/10^5;" Nw = ";c/10^5/9.8 ;" Kp"
1210 IF t2=2  THEN PRINT c; "Nw = ";c/9.8;" Kp = ";c*10^5;" Dinas"
1220 IF t2=3 THEN  PRINT c;" Kp = ";c*9.8;" Nw = ";c*9.8*10^5;" Dinas"
1230  GOSUB 9980:GOTO 9000
1300 IF t2=1 THEN  PRINT c;" Ergios = ";c/10^7;" Julios = ";c/10^7/9.8;" Kgm"
1310 IF t2=2 THEN  PRINT c;" Julios = ";c*10^7;" Ergios = ";c/9.8; " Kgm"
1320 IF t2=3 THEN  PRINT c;" Kgm = ";c*9.8;" Julios = ";c*9.8*10^7;" Ergios"
1340 GOSUB 9980: GOTO 9000
1500 ' ................. MATRIZ DE UNIDADES .............................
1505 RESTORE 1530
1510 FOR F=0  TO 3 : FOR C=0 TO 3
1520 READ U$(F,C)
1525 NEXT C,F
1530 DATA " ","C.G.S.","GIORGI","TECNICO"
1535 DATA "MASA","gr.m","Kg.m","UTM"
1540 DATA "FUERZA","dina","Nw","Kp"
1550 DATA "TRABAJO","Ergio","Jul","Kgm"
1560 RETURN
2000 '............... EJERCICIOS ....................
2005 CON=1
2010 k=INT(RND*18)+1
2020 CLS
2030  PRINT "PREGUNTA No ";CON: LOCATE 22,1: PRINT "ACIERTOS - ";AC: LOCATE 22,2
: PRINT "FALLOS   - ";FA
2035 PRINT : PRINT "-----------------------------------------"
2040  LOCATE 1,10:PRINT PRE$(K): PRINT : PRINT
2050 INPUT "RESPUESTA ";RES
2060 IF (ABS(R(K)-RES))<=0.01 THEN AC=AC+1: LOCATE 14,20: PRINT CHR$(24); " CORR
ECTO ";CHR$(24):  GOTO 2100
2070 fa=fa+1: LOCATE 14,20: PRINT "FALLO ": PRINT : PRINT : PRINT "LA RESPUESTA
CORRECTA ES ";R(K)
2100 CON=CON+1
2110  LOCATE 1,25: PRINT "PRESIONE C-Continuar, M-Menu"
2130 Z$=INKEY$:IF Z$="" THEN 2130
2140 IF Z$="C" OR Z$="c" THEN 2010
2150 IF Z$="M" OR z$="m" THEN 9000
2160 PRINT CHR$(7): GOTO 2130
2499 STOP
2500 RESTORE 2520
2510      FOR K=1 TO 18:READ PRE$(K),R(K): NEXT K
2520 DATA "Cuantos gr.m son 3 Kg.m?",3000
```



```
2522 DATA "Cuantos gr.m son 10 UTM?",9800
2524 DATA "Cuantos Kg.m son 7234 gr.m?",7.234
2526 DATA "Cuantos Kg.m son 5 UTM?",49
2528 DATA "Cuantas UTM son 9.8 Kg.m?",1
2530 DATA "Cuantas UTM son 14000 gr.m?",1.429
2532 DATA "Cuantas Dinas tiene un NW?",100000
2534 DATA "Cuantas Dinas son 2 Kp?",1960000
2536 DATA "Cuantos Nw son 9.8 Kp?",96.04
2538 DATA "Cuantos Nw son 10^7 Dinas?",100
2540 DATA "cuantos Kp son 10 Nw?",98
2542 DATA "cuantos Kp son 134567 Dinas?",.1373
2544 DATA "Cuantos Ergios son .002 Julios?",20000
2546 DATA "Cuantos Ergios son .004 Kgm?",392000
2548 DATA "Cuantos Julios son 100 Kgm?",980
2550 DATA "Cuantos Julios son 4*10^9 Ergios?",400
2552 DATA "Cuantos Kgm son 100 Julios?",10.204
2554 DATA "Cuantos Kgm son 4*10^7 Ergios?",4.0816
2560 RETURN
8000 '  PRESENTACION ..............................................
8010 PLOT 0,0
8020 FOR k=1 TO 300 STEP 10
8030 DRAWR 0,400-k: DRAWR 600-k,0
8040 DRAWR 0,-400+k+5: DRAWR -600+k+5,0
8050 NEXT k
8060 FOR K=2 TO 24 STEP 2:INK 1,K
8070 LOCATE 16,13:PRINT "UNIDADES"
8080 FOR I=1 TO 400:NEXT I
8090 NEXT K
8095 RETURN
9000 '.......... MENU ...............
9002 CLS: PRINT CHR$(24)
9005  LOCATE 16,4: PRINT "        "
9010  LOCATE 16,5: PRINT " MENU "
9020 LOCATE 16,6: PRINT "______"
9030 LOCATE 12,7: PRINT "                  "
9040  LOCATE 12,8: PRINT " 1 - GRAFICO     "
9050  LOCATE 12,9: PRINT " 2 - CALCULO     "
9060  LOCATE 12,10: PRINT " 3 - EJERCICIOS"
9070  LOCATE 12,11: PRINT " 4 - PARAR      "
9080 PRINT CHR$(24): LOCATE 11,22: PRINT "SELECCIONE OPCION"
9090 Z$= INKEY$: IF Z$="" THEN  9090
9100  X=VAL (Z$)
9105 IF X<1 OR X>4 THEN 9090
9110 ON X GOTO 100,1000,2000,50
9500 '.....  TEXTOS PARA VENTANA MOVIL...
9505 '
9515  PRINT CHR$(24)
9520  LOCATE 5,22: PRINT "--------------------------------------------------------------------------------"
9524 RESTORE 9530
9525 FOR T=1 TO 13: READ T$:GOSUB 9900:GOSUB 9960: NEXT T
9530  DATA "                         ESTE GRAFICO CONTIENE TODAS LAS INSTRUCCIONES PARA LA CONVERSION ENTRE UNIDADES DE MASA, FUERZA, TRABAJO Y POTENCIA, EN LOS TRES SISTEMAS   C.G.S  -  GIORGI  -  TECNICO "
9540 DATA "                          EL  TRIANGULO  1 QUE  ES  EL  MENOR,  REPRESENTA  LAS UNIDADES DE <MASA>: El  gramo-masa ( gr.m ) que  pertenece al sistema C.G.S., el kilogramo-masa  (Kg.m ) del GIORGI  y  la Unidad  Tecnica  de Masa ( UTM ) del TECNICO "
9560 DATA "                                    EL  2  CONTIENE  LAS  UNIDADES  DE FUERZA>:  La  dina  perteneciente  al  C.G.S.,  el Newton  ( Nw )  del  GIORGI  y  el  Kilopondio  ( Kp )  del  TECNICO.  "
```



```
9570 DATA "                                       EL   3  CONTIENE  LAS  UNIDADES  DE  <T
RABAJO   Y   POTENCIA>: El  Ergio  del  C.G.S.,  el Julio  ( Jul )   del  GIORGI
y   el  Kilogrametro  ( Kgm ) del TECNICO  "
9580 DATA "                                      Las unidades de potencia son las mism
as que las de trabajo, pero partidas por segundo, comun a los tres sistemas. El
 Jul/s. se denomina  watio  (w)."
9590 DATA "                                  LA RELACION NUMERICA ENTRE UNIDADES ES MUY
SENCILLA DE RECORDAR.  OBSERVE, QUE EN TODAS LAS BASES DE LOS TRIANGULOS, EXISTE
 UN NUMERO, 9.8,  IGUAL PARA LOS TRES.  "
9600 DATA "                                  AHORA FIJESE EN LOS SIGNOS: IZQUIERDA Y ARR
IBA  * (multiplicar), DERECHA Y ABAJO   / (dividir) "
9610 DATA "                                  POR LO TANTO, PARA PASAR DE   Kp  a  Nw  (D
rcha. a Izq.), hay que multiplicar por <9.8>  (1 Kp=9.8 Nw). DE IGUAL MANERA PAR
A PASAR DE Kgm a Julios, o de UTM a Kg.m. EVIDENTEMENTE EN SENTIDO CONTRARIO HAY
 QUE DIVIDIR. "
9620 DATA "                                  LOS LADOS DE LA IZQUIERDA DE LOS TRIANGULOS
, SE IDENTIFICAN CON UN  <10 ELEVADO A 3> EN EL INTERIOR (MASA), <10 ^ 5> EN EL
MEDIO (FUERZA) Y <10 ^ 7> EN EL EXTERIOR (TRABAJO Y POTENCIA). "
9630 DATA "                                  AHORA LOS SIGNOS A CONSIDERAR SON ARRIBA (*
) Y ABAJO (/).  POR EJEMPLO:  PARA PASAR DE   Nw  a  dinas, HAY QUE MULTIPLICAR
POR <10 elevado a 5>. DE Ergios a Julios <dividir por 10^7>, ETC. "
9640 DATA "                                  POR ULTIMO EL LADO DERECHO SE IDENTIFICA EN
 CADA TRIANGULO, POR EL PRODUCTO DE LOS OTROS DOS LADOS, CON EL CRITERIO DE SIGN
OS ANTERIOR. POR EJEMPLO DE   gr.m  a  UTM, hay que dividir por <9.8 y por 10^3>
  "
9650 DATA "                                  RECUERDE 9.8 ABAJO. 10 ↑ 3, 5, 7 A LA IZQUI
ERDA Y EL PRODUCTO A LA DERECHA. MEMORICE LAS UNIDADES Y CON EL CRITERIO DE SIGN
OS Y NO HABRA MAS CONVERSIONES  DIFICILES. "
9660 DATA "                                  UNA ULTIMA RECOMENDACION, PARA APRENDER BIE
N EL CUADRO, TRATE DE REPRODUCIRLO DE MEMORIA, HASTA QUE CONSIGA COMPLETAR TODOS
 SUS DATOS, LUEGO REALICE CONVERSIONES MENTALMENTE. LA OPCION <3> DEL MENU LE AY
UDARA. "
9890 RETURN
9900 ' ..... ESCRITURA EN VENTANA .....
9905     FOR K=1 TO LEN(T$)
9910  LOCATE 1,24: PRINT MID$(T$,K,79)
9920  SOUND 1,3,10 :     FOR Y=1 TO 60: NEXT Y
9930 NEXT K
9950 RETURN
9960  LOCATE 1,24: PRINT "            POR FAVOR, PRESIONE UNA TECLA PARA CONTINUAR
"
9970 Z$=INKEY$ : IF Z$="" THEN  GOTO 9970 ELSE RETURN
9980 LOCATE 1,24: PRINT "   PRESIONE UNA TECLA PARA CONTINUAR"
9990 Z$=INKEY$ : IF Z$="" THEN  GOTO 9970 ELSE RETURN
```

Nota: Todos los símbolos ∧ que aparecen en el listado deben sustituirse por ↑

# EDITORIAL

*Es fácil escribir cuando uno se dirige a unos amigos. Este es un momento feliz para toda la redacción de AMSTRAD EDUCATIVO. La gran avalancha de cartas recibidas nos ha demostrado el gran interés por mejoraros en el manejo de vuestro ordenador, lo cual nos ha servido de estímulo para tratar de superarnos día tras día.*

*En este número hemos incluido en el apartado técnico, el programa "Leyes de momentos de una viga", que basado inicialmente en el método de Cross nos calcula el momento resultante de la curva obtenida.*

*El juego que os ofrecemos listado este mes dentro de la sección PROGRAMANDO EN BASIC es el conocido "Juego de los Barcos". En él podéis introducir una serie de mejoras que esperamos os gusten.*

*Que os divirtáis y hasta el próximo número.*



# SUMARIO




**Edita:** Grupo Editorial G.T.S., S. A.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02.
**Director:** Antonio Bellido.
**Colaboran** en este número:
Juan M. Pintor, Víctor J. Campo López.
**Maquetación:** Susana M. Villalba.
**Secretaria de Redacción:** Mercedes Jamart.
**Publicidad:** Dpto. propio.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02

**Fotocomposición:**
Fotocom, S. A.
**Fotomecánica:**
La Unión
**Imprime:**
Gráficas FUTURA Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23
Polig. Ind. Cobo Calleja
FUENLABRADA (Madrid)
**Distribuye:**
R.B.A. Promotora de Ediciones, S. A.
Travesera de Gracia, 56 Atico, 1.ª.
Tfno 93 200 82 56
**Depósito Legal:**
M. 8.904-1986

# EL AMSTRAD Y EL CM/P

## *Referente a la denominación de ficheros en CP/M*

Todo manual de un ordenador que trabaje, o pueda trabajar, bajo el control del CP/M, tendrá un apartado dedicado a la forma de aludir, o referirse, a los ficheros (**file references**) contenidos en disco.

Por diseño, CP/M permite referirse a un sólo fichero de entre los almacenados en el disco insertado en una unidad de disco predeterminada, o a un grupo de ficheros de ese disco.

Dicho esto, recordemos que a todo fichero se le debe asignar a un nombre.

CP/M tiene establecidas unas reglas generales para dar nombre a un fichero, las cuales han de ser tenidas en cuenta en el momento de la denominación, y que son las que se exponen a continuación.

En primer lugar, el nombre debe tener al menos un caracter y, como máximo, ocho.

Por otra parte, este nombre puede tener un apéndice a su derecha compuesto por un punto y haste tres caracteres.

Este apéndice, dependiendo del manual o publicación que hable del tema, recibe diferentes apelaciones, como son "extensión de fichero", "nombre secundario" o "tipo de fichero"

Nosotros aquí, lo mencionaremos por **tipo de fichero**, ya que está más acorde con la función que cumple.

En cuanto a los caracteres de uso prohibido al dar el nombre a un fichero, y a su tipo, depende de lo que cada manual diga al respecto.

No obstante, no se deben usar en general los siguientes:

< >. , ; : = ? [ ] $

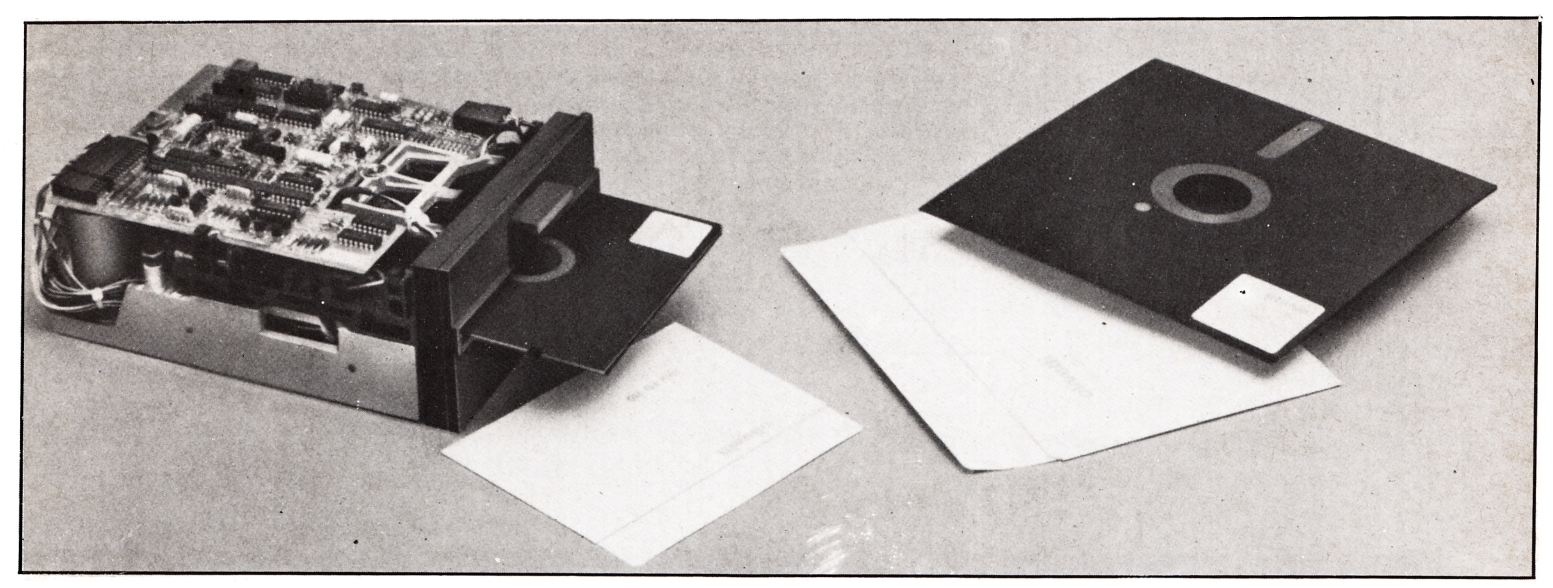
Como por ejemplo para aplicar lo dicho, supongamos que se ha de poner nombre a un fichero dedicado a controlar una biblioteca, con libros dedicados a cualquier materia. En estas condiciones, podrían valer tanto LIBROS como BIBLIOTE.

Más, si el caso fuera reducido a volúmenes conteniendo exclusivamente temas geográficos, quizá fuera conveniente usar un nombre de fichero seguido de un tipo de fichero, de este modo:

**LIGROS. GEO o BIBLIOTE. GEO**

En este sentido, es conveniente añadir que, si bien el operador puede asignar a su juicio, el tipo de fichero, algunos lenguajes y aplicaciones desarrolladas baja CP/M asignan el tipo de fichero automáticamente, y suelen responder a alguna de las denominaciones dadas en la siguiente lista:

| TIPO de fichero | Contenido |
|---|---|
| * ASC ....... | Caracteres ASCII. |
| * ASM ....... | programa en lenguaje de asamblea del tipo Z-80. |
| * A86 ....... | programa en lenguaje de asamblea del tipo 8086. |
| * BAK ....... | copia de un fichero existente con el mismo nombre. |
| * BAS ....... | programa en BASIC. |
| * CAL ....... | datos de Supercalc. |
| * CMD ...... | programa transitorio ejecutable, en CP/M-86. |
| * COB ....... | programa en COBOL. |
| * COM ...... | programa transitorio ejecutable, en CP/M-80. |
| * DAT ....... | fichero de datos. |
| * FOR ....... | programa en FORTRAN. |
| * HEX ....... | fichero hexadecimal producido por ensamblador (CP/M-86). |
| * H86 ....... | fichero hexadecimal producido por ensamblador (CP/M-80). |
| * ITN ........ | programa compilado en c BASIC. |
| * LST ....... | listado de compilación o asamblea. |
| * OBJ ....... | programa objeto. |
| * PAS ....... | programa en PASCAL. |
| * PRN ....... | listado de compilación o asamblea. |
| * REL ....... | programa objeto reubicable. |
| * SUB ....... | fichero de órdenes a ejecutar por SUBMIT. |
| * TXT ....... | textos. |
| * $$$ ........ | temporal. |



Hasta el momento, se está aludiendo a un sólo fichero por su nombre completo, sin ninguna clase de ambigüedad.

Este tipo de referencia a ficheros, se conoce en inglés como **unambiguous file name,** y de aquí el acrónimo **ufn** que será utilizado como sinónimo de "nombre fichero sin ambigüedad".

**COMODINES (WILD-CARDS)**

Veamos, a continuación, como seleccionar un ficheo o un grupo de ficheros de forma ambigüa: **ambiguorus file nafe (afn).**

Para cubrir este objetivo, CP/M pone a nuestra disposición ciertos caracteres, llamados "comodines" **(wild-cards),** de los que se da detalle a continuación:

## ?

**Función asignada:** Este comodín vale por cualquier caracter de los admitidos para conformar el nombre y tipo de un fichero.

**EJEMPLOS:**

1.º— Supongamos que se ha creado una serie de ficheros con el mismo nombre pero los tipos varían de esta forma:
BIBLIOTE. G11
BIBLIOTE. G12
BIBLIOTE. H11
BIBLIOTE. H12
BIBLIOTE. L11
BIBLIOTE. L12

En esta situación, si interesara referirse a todos los ficheros cuyo nombre fuera BIBLIOTE y su tipo comenzara por cualquier caracter y los dos últimos fueran 11, definiríamos este grupo mediante el siguiente **afn:**
BIBLIOTE. ?11
Con lo cual estaríamos aludiendo a:
BIBLIOTE. G11
BIBLIOTE. H11
BIBLIOTE. L11
.............
.............

2.º— Si ahora, con el fin de fijar la idea, admitimos, además, este otro conjunto de ficheros:
BIBCAT. G11
BIBCAT. G12
BIBCAT. H11
BIBCAT. H12
BIBCAT. L11
BIBCAT. L12

En esta ocasión, nuestro deseo es localizar todos los ficheros cuyos nombres comiencen por BIB y su tipos comiencen por H y acaben con cualquier caracter. Para ello tendríamos que dar la siguiente referencia:
BIB?????. H1?
El grupo seleccionado así es:

BIBLIOTE. H11
BIBLIOTE. H12
BIBCAT. H11
BIBCAT. H12

3.º.—Observar que no es la misma la referencia.
BIBCAT. ?1 que BIBCAT. ??1
El primero simplemente no existe. El segundo está haciendo referencia a todos los ficheros BIBCAT que acaben en 1.
Utilizando este comodín podemos aludir a todos los ficheos de un disco, mediante la referencia
????????. ???
Lo cual dará el conjunto de los ficheros contenidos en el disco.
Pero, para conseguir esto, es más cómo recurrir a otro comodín.

## *

**Función asignada:** Este comodín vale por el nombre y/o el tipo de un fichero.

**Ejemplos:**

1.º— Con todos los ficheros citados anteriormente colocados en un disco, la referencia:
* . L12
nos seleccionaría:
BIBLIOTE. L12
BIBCAT. L12

2.º— En las mismas circunstancias, la referencia:
BIBCAT . *
daría
BIBCAT... G11
BIBCAT... G12
BIBCAT... H11
BIBCAT... H12
BIBCAT... L11
BIBCAT... L12

## EJERCICIOS CON NOMBRES DE FICHEROS

1.º— ¿Es válido este nombre de fichero?: 1412DC.PRU ...
Respuesta.—No. Porque comienza con un caracter distinto a una letra.

2.º— ¿Es válido este nombre de fichero?: DONQUIJOTE ..
Respuesta.—Sí, pero sólo se aceptan los ocho primeros: DONQUIJO.

3.º— ¿Es válido este nombre de fichero?: QUIJOTE. CER .
Respuesta.—Sí.

4.º— ¿Es válido este nombre de fichero?: SEAT.127 .....
Respuesta.—Sí.

5.º— Cuál es el tipo de fichero MOVCPM.COM? ........
Respuesta.—COM.

6.º— ¿Qué ficheros quedan referenciados por *.*? ......
Respuesta.—Todos los del disco.

7.º— ¿Qué ficheros quedan aludidos mediante *.COM? .
Respuesta.—Todos los de tipo .COM contenidos en el disco.

8.º— ¿A qué ficheros aludimos con PRUEBA.12? .......
Respuesta.—A todos cuyo nombre y dos primeros caracteres coincidan y estén contenidos en el disco.

9.º— ¿Es la misma referencia * PRUEBA.*23 que *.*? ...
Respuesta.—Sí. Los caracteres a la derecha de un asterisco no son considerados.

10.º— ¿Es la misma referencia* PRU*.B* que PRU?????.B?? ? ........
Respuesta.—Sí. Los caracteres a la izquierda de un asterisco si son considerados.

## EJERCICIOS CON UNIDADES DE DISCO

1.º— Cuál es la unidad de disco por defecto si la impronta en pantalla es "A>" .......
Respuesta.—A.

2.º ¿Qué indica a CP/M la orden B:? .......................
Respuesta.—Que cambie la unidad de disco por defecto a la unidad B.

3.º ¿Cuál será la impronta tras la orden anterior? .......
Respuesta.—B>.

4.º ¿Es correcta esta orden? Ñ:
Respuesta.—No, porque la Ñ no es caracter estandar ASCII y, además, y en todo caso, la versión 2.0 sólo permite en línea 16 unidades de disco.

### REFERENTE A LAS UNIDADES DE DISCO

Páginas atrás comentábamos que las unidades de disco conectadas a un computador son reconocidos por el sistema mediante letras.

Estas letras, suelen ser A, B, C y D, aunque cada fabricante impone sus criterios. Por esta razón, se deberá confirmar este punto en el manual de cada ordenador. Aquí se seguirá el criterio anterior.

También se dijo, que inmediatamente después de la conexión, la unidad de disco elegida por CP/M es la A. Esta situación queda reflejada en la pantalla al mostrar
A>

Como sabemos, esta es la impronta indicativa de que CP/M está a nivel operador o, dicho de otro modo, el CCP está a la espera de órdenes.

Pero, en estas circunstancias, todas las órdenes CP/M relativas a ficheros serán sobreentendidas y dirigidas a la unidad de disco A, y al disco instalado en ella.

Esta es la "unidad por defecto" o "unidad implícita".

El operador, cuando la CP/M está a su nivel, puede dar la orden oportuna para cambiar a otra unidad de disco, y convertirla, así, en la unidad de disco por defecto actual.

Esta orden consiste en escribir la letra correspondiente a la unidad de disco seleccionada seguida de dos puntos (:). Ejemplo:
A>B:

Al pulsar <c r>, la pantalla indicará el cambio de esta forma

A>B:
B>

A partir de este momento, todas las referencias a ficheros serán dirigidas al disco colocado en la unidad B.

Cerraremos este epígrafe diciendo que todas las minísculas introducidas al nombrar un fichero o seleccionar una unidad de disco son transformadas a mayúsculas, al ser procesadas por CCP.

# COMO CREAR FICHEROS EN DISCO

Cuando se introduce el disco en la unidad de disco, la corredera indicada con la letra F en la figura 1 de desplaza hacia atrás, permitiendo a la cabeza magnética encargada de la lectura-escritura en la superficie del disco, tomar contacto con ella a través del taladro alargado C.

El cabezal se mueve radialmente aldelante y atrás a lo largo de esta abertura, para poder acceder a cualquier punto del disco.

En este sentido debemos resaltar la gran rapidez de maniobra que implica el doble movimiento de giro del disco y radial de la cabeza, con el consiguiente ahorro de tiempo en la lectura-escritura de información.

D) * Los taladros denominados de alineación son los encargados de ajustar la carcasa del disco en su correcta posición dentro del *disk-drive,* haciéndolo, de esta forma, solidario con él.

E) * Finalmente, el taladro de índice permite al dispositivo adecuado de la unidad de disco «ver» dónde comienzan los sectores de las pistas, mediante una pequeña perforación existente en la superficie del disco. El dispositivo en cuestión espera hasta detectar la perforación citada y, a partir de aquí, lleva la cuenta de los **bytes** de información.

Los conceptos de sector y pista se estudian más abajo.

Veamos ahora la forma en que se organiza el almacenamiento de la información sobre la superficie magnética del disco, y qué y cómo lo maneja.

Al comprar un disco flexible o compacto, lo que realmente hemos adquirido es una lámina de plástico circular, recubierta de un material susceptible e ser magnetizado, ofreciéndonos su superficie para ser usada como soporte de información. Algo de eso se dijo la principio de este epígrafe, pero se omitió que, por lo demás, no sirven para nada... hasta que se les somete el proceso de **formateado.**

La expresion **formateado**es una anglicismo derivado de la voz inglesa **format**, con lo cual se quiere dar a entender que, mediante éste, el disco va a adquirir determinadas características. FORMAT es un programa de utilidad tratado en otro lugar detalladamente.

Bien. Pero, ¿qué ha cambiado en el disco, después del formateado?

## ESQUEMA DE UN DISCO FORMATEADO POR UN AMSTRAD

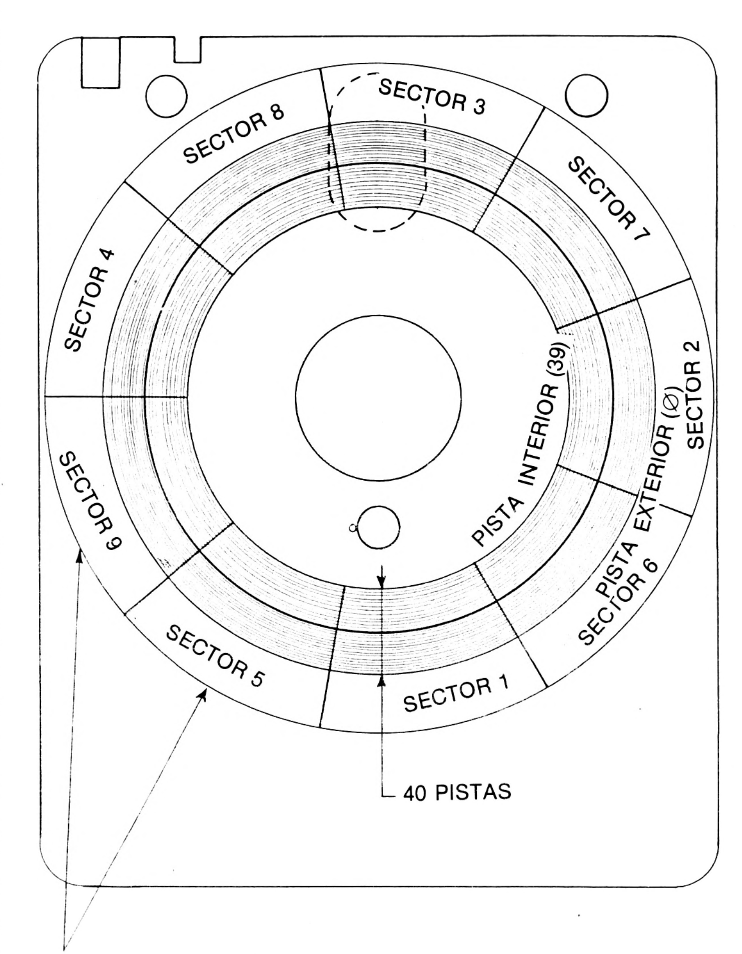



Figura 2: DISCO FORMATEADO (INTERIOR)

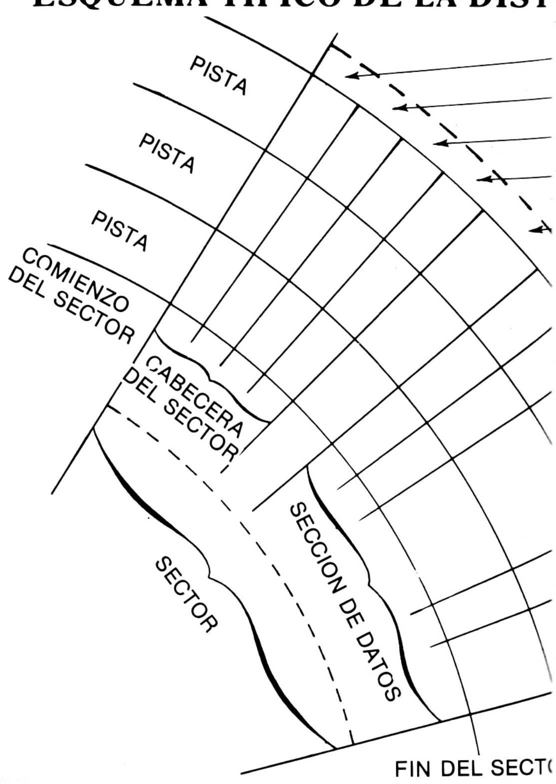
Resumen para un disco formateado por un AMSTRAD.

* 40 pistas. Numeradas de 0 (exterior) a la 39 (interior)
* 9 sectores por pista ><360 sectores por cara.
* 512 **bytes** de espacio para datos por sector = 4608 bytes pora pista ><4,5 bytes.
* 64 ficheros, como máximo, se pueden almacenar en un disco, pero aquellos ficheros que ocupen más de 16 K bytes reducirán este número en una unidad por cada 16 K.
* 1024 bytes que, como mínimo, ocupará un fichero. También denominado tamaño del bloque.
* La capacidad del disco variará de 180 K bytes a 171 K bytes, según el disco se formatee con los comandos CP/M incluidos, o sólo sea utilizados para datos.

**ESQUEMA TIPICO DE LA DISTRIBUCION LONGITUDINAL DE UN SECTOR**

1.— Comprueba si la velocidad de rotación es la adecuada.
2.— Identifica pista.
3.— Indentica sector.
4.— Comprueba número de bytes grabados en la cabecera (check-sum).
5.— Longitud en blanco para separar cabeza.
6.— Similar a 1.
7.— Idenfica siguiente sector.
8.— Información almacenada.
9.— Similar a 4 pero relativo a la Sección de datos.
10.—Longitud en blanco para separar el Sector del siguiente.

**ESQUEMA TIPICO DE LA DIST**




Para describir el cambio supondremos que estamos trabajando en una máquina específica:el AMSTRAD.

En primer lugar, la cabeza magnética de la unidad de disco escribe cierta información, borrando cualquier otra que pudiera existir, en una serie de anillos circulares denominados **pistas.** El AMSTRAD configura 40 de estas pistas, numeradas del 0 (la más externa) al 39 (la más interna).

Cada **disco** a su vez, es dividido en un número predeterminado de sectores circulares de igual superficie entre sí.

Por extensión denominamos **sector** a los diferentes trozos de pista comprendidos en un sector (bloque). El AMSTRAD los fija en 9, numerados del 1 al 9, y distribuidos según indica el dibujo 2.

En la figura 3 se puede ver un esquema típico de la información depositada en un sector cualquiera, después del formateado.

De toda ella lo que más nos interesa es la longitud disponible para datos, equivalentes a 512 bytes en el AMSTRAD. Físicamente aquí es donde se "memoriza" la información procedente del computador y recibida a través de la oportuna transmisión; el resto del espacio del sector está destinado a contener la información que permitirá al DOS el control sobre todos y cada uno de los sectores, y, finalmente, el disco en su calidad de soporte externo de memoria.

Observese, en el dibujo 2, que tras el formateado, sólo un tercio de la superficie de cara cara es utilizada.

En el AMSTRAD, en términos de transmisión entre memoria RAM y disco, la unidad más pequeña de información que se puede manipular es un sector o, lo que es lo mismo, **512 bytes.** Para trabajar a nivel de bytes hay que pasar del disco a RAM el sector que le contenga y allí manipularlos.

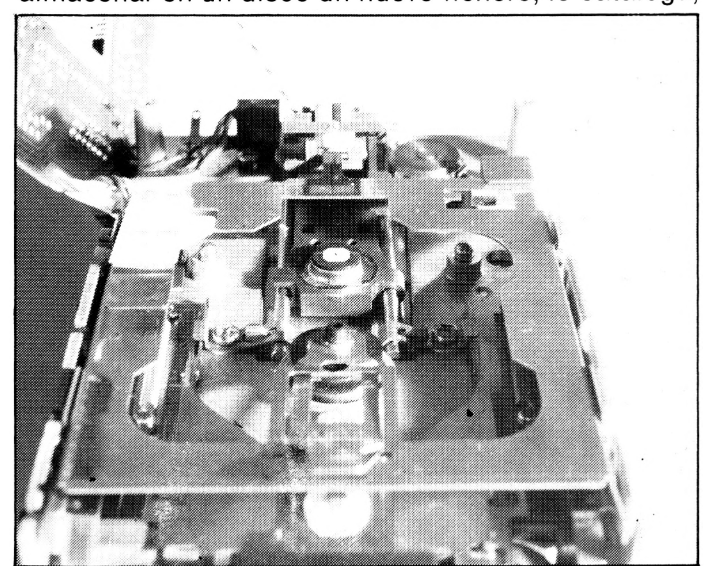
## DISK-DRIVE

Con estos conocimientos en nuestro haber, vamos a aproximarnos al dispositivo periférico como **disk-drive** o unidad de disco.

Dediquemos una líneas a la **pista catálogo o directorio.**

Esta pista suele ser la central, como muestra la figura 2, o la número 2 contada de sde el exterio (0), y su misión es, contener la información necesaria y suficiente a disposición del DOS, para que este pueda, en su momento y según los casos, organizar los procesos de lectura y escritura en el propio disco. En MS-DOS, el directorio ocupa los sectores 4, 5, 6 y 7 de la pista ∅.

El Sistema operativo, para cumplir esta misión, al almacenar en un disco un nuevo fichero, lo cataloga,



## RIBUCION LONGITUDINAL DE UN SECTOR

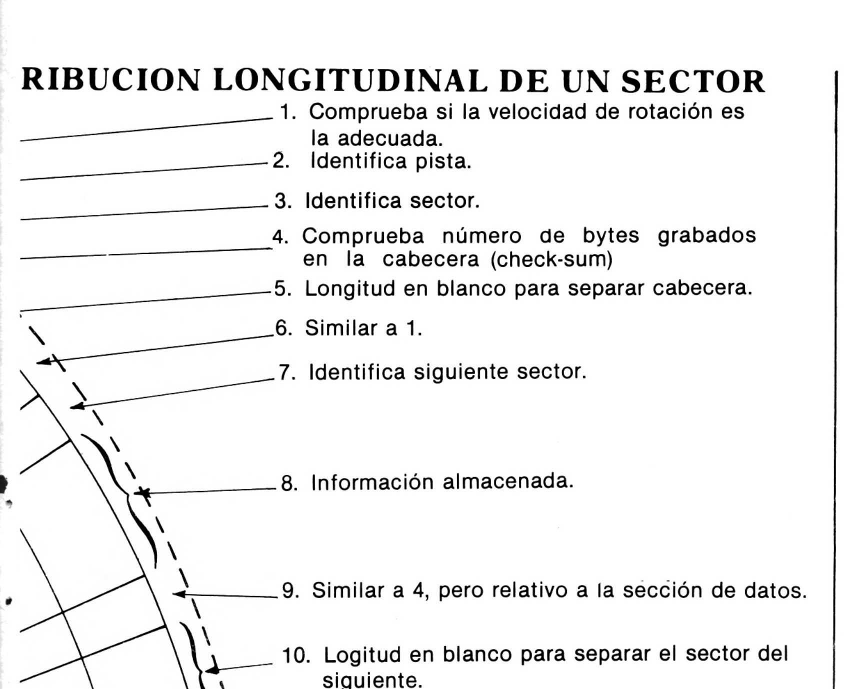



Figura 3: Esquema típico de en sector en CP/M.

grabando en la pista directorio una secuencia de treinta y dos **bytes**, con lo cual, y entre otras cosas que se verán con detalle más adelante, se deja constancia del nombre del fichero y de la posición y espacio que ocupa.

En el lenguaje cotidiano, inglés, la estructura mental de la palabra **drive**, soporta significados tales como **conducir, transmitir** e **impulsar**, por tanto, en términos informáticos, deberíamos interpretar que un **disk-drive** es un dispositivo para transmitir datos entre un computador y un disco, y, además, maneja —conduce e impulsa— el propio disco, para lo cual está dotado de la electrónica y mecánica necesaria.

Así pues, un disk-drive es un dispositivo electromecánico que permite el intercambio de información entre el computador al cual está conectado, y un disco instalado en él.

Nosotros nos referiremos a este periferico como a una unidad de disco.

Dentro de la unidad de disco, no podemos decir que haya una parte más importante que otra ya que del preciso y acorde trabajo de todas ellas, depende la eficacia de este periférico.

No obstante, la cabeza magnética parece la pieza del sistema y a ella nos referiremos en primer lugar.

Una cabeza lectora-grabadora es, esquemáticamente, un anillo metálico con una abertura en un lugar, y en el opuesto una bobina. Cuando por esta bobina pasa una corriente eléctrica, en la abertura se genera un campo magnético cuya intensidad variará conforme a determinadas alteraciones en las características de la corriente que circule por la bobina.

En sentido contrario, si en las proximidades de la abertura se producen variaciones magnéticas, en la bobina se genera una corriente eléctrica.

Al hacer girar un disco en el **disk-drive**, estamos poniendo a disposición de un artificio capaz de magnetizar, tal cual es la cabeza de lectura/escritura, una superficie susceptible de ser magnetizada.

Cuando se pretende grabar —escribir— alguna información en el disco, lo que hacemos, mediante las instrucciones oportunas es obligar al computador a emitir una serie de impulsos eléctricos —bits— que, en función de sus características —ausencia/presencia de energía—, producen en la cabeza variaciones del campo magnético que quedan registradas en la superficie del disco.

Si, por el contrario, se desea recuperar la información almacenada en un disco, le pondremos a girar para que, al pasar enfrente de la cabeza, las partículas que componen la superficie magnética de la cara en la cual se quiere leer —magnetizadas con diferente intensidad en el proceso de grabación explicado anteriormente—, produzcan una corriente eléctrica cuyas variaciones serán transmitidas al computador.

Con el fin de estas variaciones respondan a un mismo criterio, es imprescindible que el disco gire a una velocidad constante, lo cual se consigue gracias a una electrónica y una mecánica sofisticadas.

Si estas variaciones alcanzan ciertas características estamos en presencia de un 1, en particular otro caso de un 0.

Por otra parte, y como ya sabemos, tanto el cero como el uno, son **bits** o unidades elementales de información y, una sucesión de bits, almacenados en forma de **bytes,** es, en definitiva, el contenido de un disco.

La cabeza de lectura está situada en un cabezal que se desplaza hacia delante y hacia atrás en la dirección del radio del disco, y a través del taladro alargado que a este fin tiene la funda de esta. El motor encargado de este movimiento es de extrema precisión consiguiendo desplazamientos, sobre la superficie del disco, muy pequeños y calculados.

En el caso de intentar la grabación de una información en disco, la primera acción que lleva a cabo el cabezal, es desplazarse hasta situar la cabeza sobre la pista catálogo, para que aquella "lea" el contenido de ésta y, determine si hay algún fichero con el mismo nombre y sitio en el disco.

La pista catálogo, contiene todos los datos necesarios para controlar la información que, hasta el momento esté grabada en el disco.

En el supuesto de haber espacio suficiente, se inicia la grabación en un sector vacío y si son necesarios más se utilizan nuevos sectores. Los datos oportunos para controlar esta nueva información a grabar, son escritos en la pista catálogo y en los propios sectores.

Un proceso similar se utiliza para recuperar la información contenida en el disco. Una vez localizada la pista y sector donde comienza, se lee esta, y si ocupa más de un sector se pasa al siguiente hasta completar la lectura de todo el espacio ocupado.

En el proceso de borrado de una información, los datos de esta desaparecen de la pista catálogo, dando por libres los sectores que estuvieran ocupados.

Todas estas funciones, y otras muchas, están dirigidas por el DOS.

# EL BASIC DE

## COMO SE LISTA UN PROGRAMA

Hacer, o sacar, un listado de un programa ya introducido en el computador, como es el caso del ejemplo anterior, consiste en hacer imprimir en pantalla todas las líneas que lo componen y en el orden de menor a mayor, según sus números.

El BASIC dispone del comando LIST que cumple esta función; más adelante se verá en profundidad, de momento vamos a aplicarlo al pequeño programa que hemos tecleado. Escribe LIST seguido de ENTER y verás aparecer el listado sin la línea 20.

## COMO SE EJECUTA UN PROGRAMA

El pequeño programa anterior —actualmente con dos líneas: la 10 y la 30— está en memoria, pero nada le obliga al computador a ejecutarlo, o, dicho de otro modo, a obedecer las instrucciones que lo componen.

El comando BASIC encargado de esto es RUN, del cual anticipamos lo dicho con objeto de ver el efecto que causa sobre nuestro ejemplo.

Escribe RUN seguido de ENTER.

Lo primero que salta a la vista es el mensaje de error de sintaxis que emite el computador, refiriéndose a la línea 30, la cual, por otra parte, aparece completa a renglón seguido, y con el cursor a continuación del número de línea.

Esto es lógico si consideramos que el comando BASIC correcto es PRINT y no PRRINT. La acción lógica subsecuente por nuestra parte será borrar una de las erres que sobran; para ello desplacemos el cursor, mediante la teclar marcada >, hacia la derecha y hasta colocarlos en la posición elegida. Ahora, basta con apretar la tecla marcada CLR para que el carácter, sobre el cual está el cursor, desaparezca.

Pulse ENTER y pida un nuevo listado (LIST y ENTER).

Verá que el defecto está subsanado.

Ejecute el programa (RUN y ENTER) y el programa será obedecido sin más problemas. De una forma deliberada hemos introducido otros errores, ahora relativos a las primeras frase de el Quijote. Así, por ejemplo, en la línea 10 nos sobra una r en lugarr. Teclee EDIT 10 seguido ENTER y verá aparecer la línea 10 con el cursor sobre el primer carácter. Desplace el cursor con la tecla adecuada (>) y proceda como en el caso anterior.

En la línea 30, tenemos una doble modificación, ya que por una parte hay que añadir **no** entre **nombre** y la palabra que la sigue, y, por otra, debemos separar **quiero** de **acordarme.**

Empecemos por editar la línea 30 (EDIT 30 y ENTER). Cuando la línea en cuestión haya aparecido, desplacemos el cursor hasta colocarlo sobre la q, y una vez allí teclea no y un espacio. Después coloca el cursor sobre la primera a de **acordarme** y da un espacio. Cuando todo ha sido hecho, pulsa ENTER y pide un nuevo listado. Comprobarás que todas las correcciones han sido efectuadas.

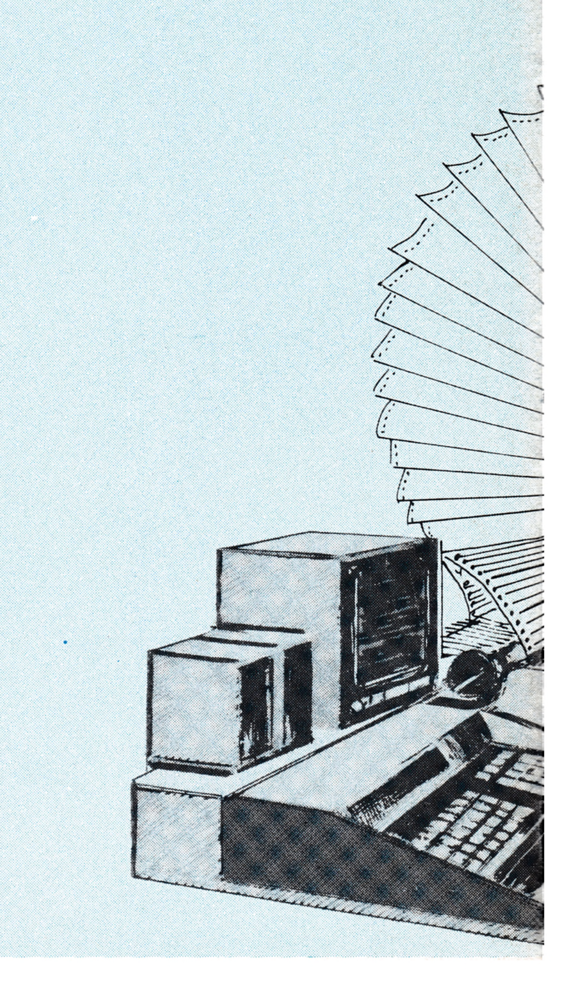
Hay otra forma de corregir listados, basada en el cursor de copia. Para ver esto en acción, supongamos nuevamente las tres líneas iniciales del programa ejemplo, errores incluidos.

Cuando hayas terminado de teclear el programa, el cursor estará situado al comienzo de la siguiente línea disponible.

Comencemos las correcciones, por el nuevo sistema, en la línea 10.

Manteniendo SHIFT apretada, pulse la tecla > las veces que necesite como para situar el nuevo cursor que aparece —cursor de copia— sobre el 1 del comienzo de la línea 10. Ahora aprieta la tecla donde figura COPY y verás desplazar este cursor sobre los caracteres de esta línea y, simultáneamente, el cursor normal se desplazará dejando aparecer los mismos caracteres. Para sobre la siguiente posición al carácter que desees borrar, la r en este caso, y pulsa DEL. Finalmente corre el cursor (con COPY) hasta el final de la línea y aprieta ENTER. La corrección ha sido efectuada.

Atención. Si el cursor de copia no se desplaza hasta el final de la línea, la parte de la línea situada a la derecha de este cursor, queda borrada al pulsar ENTER.



Es evidente que, para borrar una línea completa, es más cómodo el primero de los sistemas propuestos, no obstante, este es un ejercicio interesante ya que, al desplazar el cursor de copia hasta el extremo derecho de la línea 20, y proceder a su borrado mediante DEL, si el cursor llega a "eliminar" los números de línea, el proceso se cancela y la línea se mantiene, en otras palabras, sólo se pueden borrar las sentencias, o parte de ellas, situadas en una línea, nunca el número de línea.

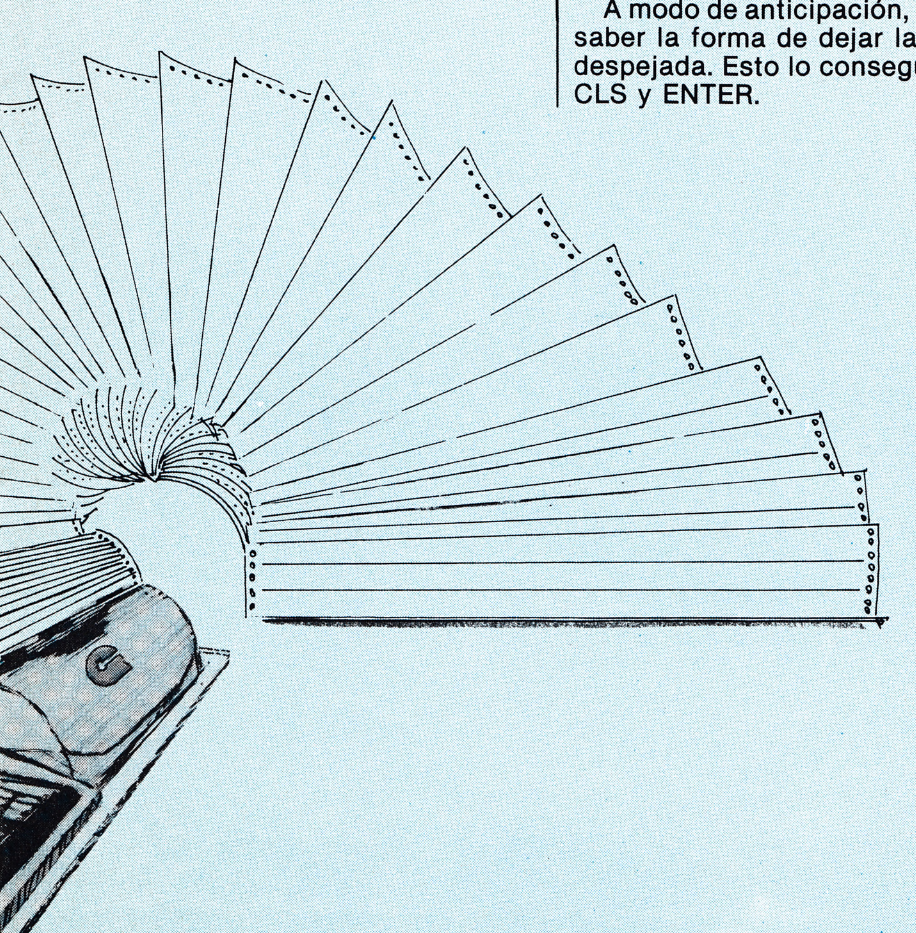
También debemos observar que la tecla CLR no es operativa con el cursor de copia.

Prueba a efectuar las modificaciones de la línea 30 mediante el cursor de copia, siguiendo las indicaciones anteriormente dadas.

**COMO SE BORRA UN PROGRAMA EN MEMORIA Y COMO SE LIMPIA LA PANTALLA**

Una vez que hayas hecho tus pruebas con el programa anterior y si, por alguna razón, deseas tener la memoria libre de cualquier listado, puede usar el comando NEW (seguido de ENTER, claro está), con la cual, si pruebas a pedir un listado, verás que no hay tal.

A modo de anticipación, debemos saber la forma de dejar la pantalla despejada. Esto lo conseguirás con CLS y ENTER.



**CONEXION Y PUESTA EN MARCHA**

Para hacer operativo tu sistema de computación AMSTRAD, necesita conectarlo correctamente.

**Mantener el equipo desenchufado de la red mientras procedes a interconectarlo y siempre que no esté en uso.**

El primer paso debe consistir en comunicar el computador con el monitor —o su televisión.

Las conexiones encargadas de esto, están en la parte posterior de la carcasa y a la altura del teclado numérico y marcadas con:

MONITOR
5 V DC
12 V DC

Tanto en el supuesto de que dispongas de un monitor AMSTRAD, como en el caso de que, desees utilizar la pantalla de tu TV, a través de un modulador, cada clavija tiene una sola conexión posible. Si se trata de monitor bastará con unir computador y monitor mediante los tres cables que, de uno u otro dispositivo emergen, conectar el monitor a la red y apretar el botón POWER situado en la esquina inferior izquierda del frontal de la pantalla. Si se usa el modulador, la explicación anterior es válida, pero relativa a este dispositivo; habrá claro está, que unir el modulador con la TV mediante el cable dirigido a la entrada de antena del televisor.

Dado que el transformador está situado en el interior del modulador, la toma de energía se efectuará conectando éste a la red.

En función del modelo de AMSTRAD de que se disponga, siempre existirá la opción de conectar un magnetófono a cassettes y un "floppy". Los lugares de conexión son evidentes y no requieren mayor explicación.

Las impresoras que de forma directa, se pueden utilizar deben incor-

porar una **interface** centronics. Su conexión en el computador está indicada con **PRINTER** en la parte posterior.

La función que cumple cada elemento del ordenador es la siguiente:

**Computador:** EL AMSTRAD contiene toda la electrónica necesaria para el manejo y almacenamiento de la información y posee la adecuada conexión, interna, al teclado.

**Teclado:** Gracias al teclado se consigue una comunicación directa entre el usuario y el ordenador, introduciendo información a través de él. Este dispositivo es sólo de entrada.

**Televisor** o **monitor:** Este dispositivo es sólo de salida, ya que permite al computador emitir sus respuestas por la pantalla del mismo.

**Cassette** o **disquete:** Es un medio de almacenamiento externo que permite transferir información desde una cinta magnética al computador y, en sentido contrario, desde el computador a la cinta.

**Impresora:** Es un dispositivo de salida que permite al usuario proveerse de copias en papel de información contenida en el computador.

## ALGORITMO

Un algoritmo es un conjunto de operaciones perfectamente definidas, gracias a las cuales se puede resolver un problema partiendo de unos datos conocidos.

Por ejemplo: Para hallar el área de un triángulo hay que multiplicar lo que mide la longitud de su base (b) por lo que mide la longitud de su altura (a) y dividir el resultado por 2. Esto mismo expresado en forma esquemática del proceso que conlleva es:

1.° R1 = b*a
2.° R2 = R1/2

Estas son las dos operaciones (o pasos) que exige el algoritmo propuesto.

## CONSTANTES

Las **constantes** son aquellos elementos de un algoritmo cuyo valor permanece invariable.

En términos de programación hay dos clases de constantes: **numéricas** y **alfa numéricas.**

Las constantes numéricas pueden ser cualquier valor numérico positivo o negativo, entero o decimal. Como por ejemplo 3 ó —33,33.

Las alfanuméricas están compuestas por cualquierconcatenación de caracteres escritos entre comillas.

EJEMPLOS: "AMSTRAD"
"24 de diciembre"

## VARIABLES

Las variables son aquellos elementos de un algoritmo capaces de tomar cualquier valor.

En programación hay dos tipos de variables: **numéricas y alfanuméricas.**

Las primeras quedan representadas por cualquier cadena de caracteres con la única condición de que el primero de todos ellos sea una letra. EJEMPLOS: LET A = 1 o LET A2 = 1520.

Las alfanuméricas o de caracteres deben ser representadas por una cadena seguida del símbolo **$**, para que el computador la distinga, de las numéricas.

El valor que se asigne a un variable alfanumérica, debe ir entrecomillas.

EJEMPLO:
LET A$ = "AMSTRAD" o LET R$ = "24 de diciembre"

Denominamos genéricamente por nombre de la **variable** a la representación de las mismas, así, en los ejemplos anteriores, tenemos los siguientes nombres de variables:

A, A$, A$ y R$.

## OPERADORES

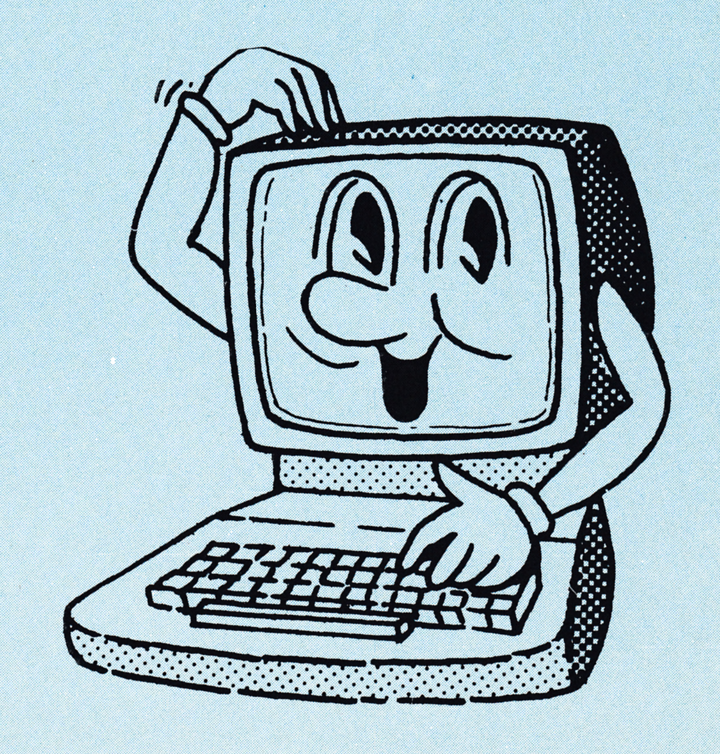
Los operadores son los símbolos que determinan las operaciones que se han de realizar entre valores.

Hay cuatro tipos de operadores: **Aritméticos, de relación, lógicos, y funcionales.**

## EXPRESIONES

Una **expresión** es un conjunto de valores relacionados entre sí por medio de operadores, cuyo resultado final es una cantidad.

Con respecto a las variables se dijo que eran elementos capaces de



contener un valor, una expresión, sin embargo, **es** un valor.

EJEMPLO: Definir una expresión que determine el valor de un viaje en taxi, sabiendo que cada "salto" del contador cuesta 25 Ptas.

Si admitimos que la variable X representa el número de "saltos" de contador, el importe del viaje vendrá dado por la expresión: 25*X.

El operador que relaciona ambos valores es "*".

Claro está, para cada valor de X tendremos un nuevo valor de la expresión 25*X.

Las expresiones matemáticas se escriben en BASIC prácticamente igual que con lápiz y papel, pero con las variables impuestas por los símbolos que figuran en el teclado del ordenador. Tal es el caso, por ejemplo de $X^3$ (equis elevado a 3), que nos obligará a teclear x↑3.

El orden de prelación de las diferentes operaciones matemáticas se verá más tarde, pero, antes de llegar al mismo, es importante hacer obser- unas sutiles diferencias que existen entre escribir una expresión matemática a mano o en un computador. Supongamos

que deseamos transcribir la expresión $\frac{a + b}{2c}$ de forma entendible para el ordenador. para lograrlo, no debemos olvidar ningún signo —como * entre 2 y c en el denominador— y no magnificar la capacidad de la máquina ya que:

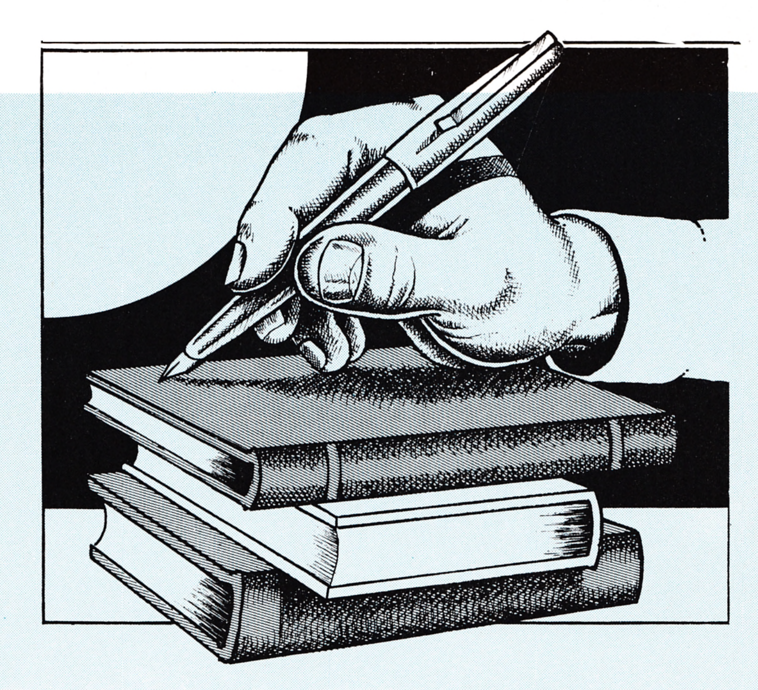
— Si escribimos a + b/2*c, el ordenador interpretará $\frac{a + b}{2}$ c.

— Si escribimos (a + b)/2*c, el ordenador interpretará $\frac{a + b}{2}c$

— Sólo podrá interpretar correctamente la expresión citada si escribimos (a + b) / (2*c).

## SENTENCIAS

Una sentencia es una frase, escrita conforme al vocabulario y sintaxis de lenguaje de programación, que transmite al computador una orden.



## OPERADORES ARITMETICOS

Se denominan así porque actúan entre valores aritméticos, siendo el conjunto de los símbolos que los componen y su orden de prioridad establecidos por el ordenador los siguientes:

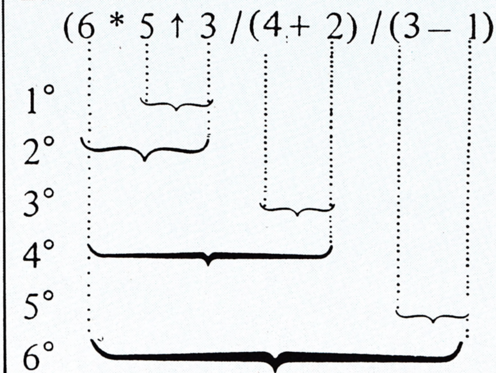
| Prioridad | Operación | Operador |
|---|---|---|
| Máxima | exponenciación | ↑ |
| | Módulo | MOD |
| | multiplicación y división | */ |
| | división entera | \ |
| | suma y resta | + — |
| mínima. | | |

Para operadores de la misma prioridad, la máquina ejecuta las operaciones de izquierda a derecha.

Las operaciones dentro de un paréntesis las efectúa primero y siguiendo el orden de prioridad anterior.

Con MOD se obtiene el resto de una división.

**EJEMPLO:**




Calcula la potencia 5 ↑ 3 ($5^3$).
Multiplica el resultado por 6.
Calcula 4 + 2.
Divide el producto (2º) entre la suma (3º).
Calcula 3 - 1.
Divide el producto (2º) entre la diferencia (6º).

**EJEMPLO:**

**PROGRAMA:**

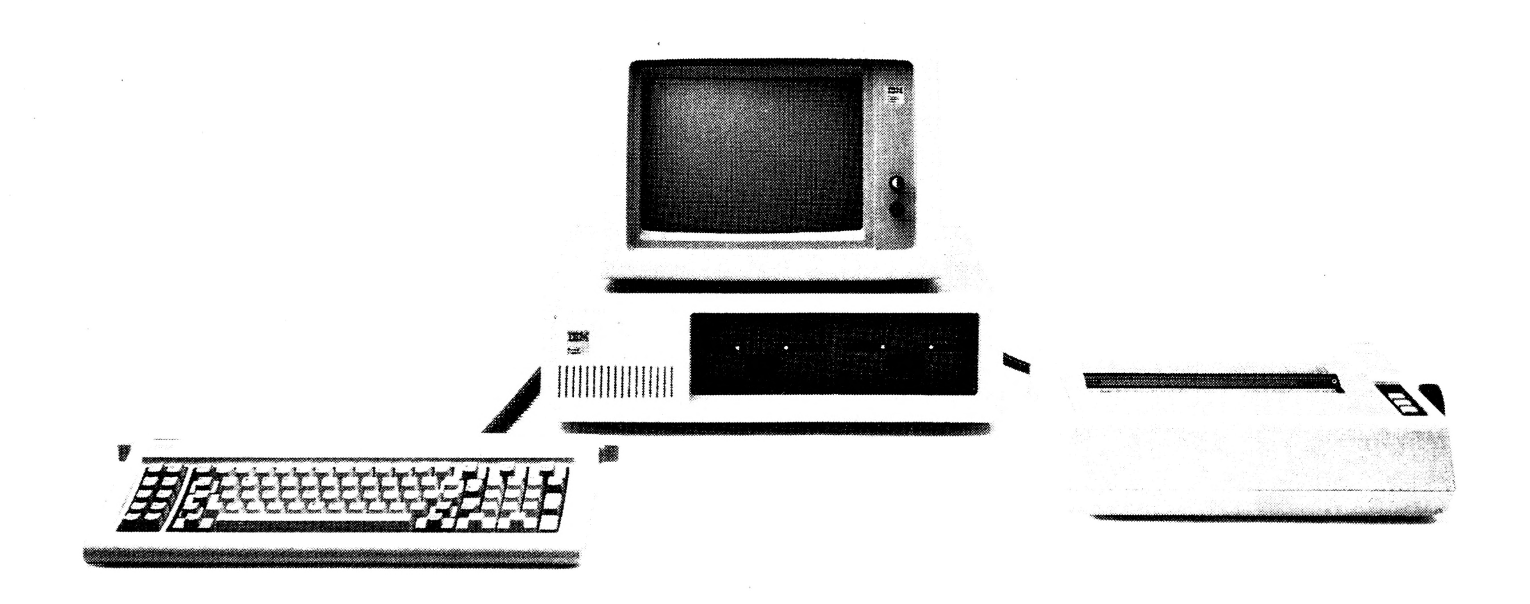
```
10 LET A$ = "ABC"
20 LET B$ = "DEF"
30 PRINT A$ + B$
```

## COMENTARIO

Con las líneas 10 y 20 asignamos valores a las variables de caracteres A$ y B$.

Con la línea 30 ordenamos la impresión del contenido actual de A$ y B$, justamente uno a continuación del otro.

El resultado de ejecutar este programa será: ABCDEF.



## OPERADORES DE RELACION

Estos operadores sólo actúan en proposiciones entre dos operandos, cuyo resultado únicamente puede ser CIERTO, que se representará por 1, o FALSO, que se representar por 0.

Sus símbolos son: igual que ... =
distinto que ................ ><
menor que ................... <
mayor que.................... >
menor o igual que........... < =
mayor o igual que ........... < =

Si un operador de relación compara dos expresiones en las que intervienen operadores aritméticos, antes de efectuar la comparación, actúan los operadores aritméticos.

**EJEMPLO:**

Determinar si 2 + 5 es menor que $(15 + 4)^3$ / 20250.

*SOLUCION:*

Es evidente que, sin efectuar las operaciones aritméticas que determinan el valor de cada expresión, es difícil responder a la proposición anterior.

Las cadenas de caracteres pueden ser comparadas con estos mismos operadores. Para hacerlo se cotejan, carácter a caracter, ambas cadenas, valorando cada caracter de acuerdo con su código ASCIL. Más adelante se estudiará todo lo referente a los códigos, baste por ahora saber que cada caracter tiene su propio valor.

## OPERADORES LOGICOS

Responden directamente a las funciones lógicas del álgebra de BOOLE.

Los **operadores lógicos** actúan entre operandos en los que, a su vez, intervienen operadores de relación. Esto quiere decir que los operandos de los operadores lógicos tendrán siempre el valor 1 (CIERTO) o 0 (FALSO) y, por consiguiente, los **operadores lógicos** lo responden 1 ó 0 (CIERTO o FALSO).

Los **operadores lógicos** NOT, AND y OR que se estudiarán con detalle más adelante.

## OPERADORES FUNCIONALES

Estos son algoritmos que residen en el propio BASIC y operan sobre datos suministrados al computador por medio del teclado o el programa, obteniéndose, como resultado un valor.

## COMA FLOTANTE

El ingeniero español Leonardo Torres Quevedo fue el primero en aplicar el concepto de la **coma flotante** a los computadores.

Cualquier número, positivo o negativo, puede ser representado en forma exponencial, para ver esto, supongamos que al medir una distancia escribimos 525 metros.

Evidentemente, podríamos haber representado lo mismo con: $0.525*10^3$ y esto, en definitiva, es lo que representa el concepto de la coma flotante.

EJEMPLO: PRINT 124* 1000 000

Al ejecutar esta instrucción en modo directo el ordenador nos retorna 124000000.

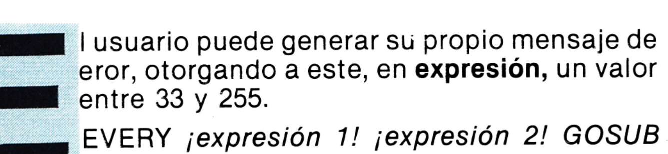
El mismo resultado habríamos obtenido con PRINT 1.24 E 8.

La mantisa queda representada por 1.24 y el índice por 8.

Este sistema permite, además de un medio económico de codificación, un amplio campo de representación numérica.

# FICHAS DEL AMSTRAD

| CODIGO | TIPO DE ERROR | COMENTARIOS |
|---|---|---|
| 1 | Unexpected NEXT | NEXT inesperado |
| 2 | Syntax Error | Error de síntasis |
| 3 | Unexpected RETURN | RETURN inesperado |
| 4 | DATA exhausted | DATA agotados |
| 5 | Improper argument | Argumento impropio |
| 6 | Overflow | Desbordamiento en cálculo |
| 7 | Memory full | Memoria llena |
| 8 | Line does not exist | Línea inexistente |
| 9 | Subscript out of range | Subíndice inadecuado |
| 10 | Array already dimensioned | Matriz ya dimensionada |
| 11 | Division by zero | División por cero |
| 12 | Invalid direct command | Comando incorrecto |
| 13 | Type mismatch | Tipo inadecuado |
| 14 | String spacefull | Espacio para cadenas completo |
| 15 | String too long | Cadena demasiado larga |
| 16 | String expression too complex | Cadena demasiado compleja |
| 17 | Cannot CONTinue | No puede continuar |
| 18 | Unknown user fuction | Función usuario desconocida |
| 19 | RESUME missing | Se alcanza el fin del programa antes del resumen de errores. |
| 20 | Unexpected RESUME | Resumen inesperado |
| 21 | Direct command found | Encontrado comando directo |
| 22 | Operand missing | Falta operando |
| 23 | Line too long | Línea demadiado larga |
| 24 | EOF met | Fin de fichero encontrado |
| 25 | File type error | Error en el tipo de fichero |
| 26 | NEXT missing | FOR sin NEXT |
| 27 | File already open | Fichero ya abierto |
| 28 | Unknown command | Comando desconocido |
| 29 | WEND missing | WHILE sin WEND |
| 30 | Unexpected WEND | WEND inesperado |
| 31 | File not open | Fichero no abierto |



El usuario puede generar su propio mensaje de eror, otorgando a este, en **expresión,** un valor entre 33 y 255.

EVERY *¡expresión 1! ¡expresión 2!* GOSUB *¡número!*

Llama a la subrrutina situada en la línea **número,** cada vez que el tiempo dado por **expresión 1** se consume.

Cuatro instrucciones EVERY pueden trabajar simultáneamente con la prioridad que determina **expresión 2,** siendo la máxima 3 y la mínima 0. La unidad de tiempo es 0,02 segundos.

```
 5 INPUT x: EVERY 100/x, PI/PI GOSUB 40.
 10 EVEREY 100,2 GOSUB 50
 20 PRINT "2"; GOTO 20
30 END
40 PRINT "####": RETURN
50 PRINT "$ $ $ $"; SOUND 1,30: RETURN
```

**FILL ¡expresión!**

Rellena una zona delimitada de gráficos de color dado por **expresión.**
Si la zona no está perfectamente cerrada, el rellenado continúa.
10 MODE 1: MOVE 0,50: DRAW 637,50
20 MOVE 0,0: x = 2
30 FILL 28/x

**FIX ¡(expresión)!**

Elimina los decimales de **expresión**
? FIX (PI)
3.

**FOR ¡variable = expresión 1! TO ¡expresión 2! STEP ¡expresión 3!**

Establece un bucle entre la línea donde aparece esta instrucción y la instrucción NEXT, el cual se repite desde que la **variable** toma el valor dado por **expresión 1,** hasta que alcanza el dado por **expresión 2,** con la cadencia determinada por **expresión 3.**

10 x = 10
20 FOR y = 100 / x TO 10* x STEP 20/x
30 PRINT y;
40 NEXT

**FRAME**

Armoniza el desplazamiento de un móvil por la pantalla.

**FRE** (expresión)

FRE (" ")
Esta función guarda la memoria libre disponible para el BASIC.

**GOSUB ¡número!**

Dirige la ejecución del programa a la subrrutina situada en la línea cuyo número es **¡número!**

**GOTO ¡número!**

Dirige la ejecución del programa a la línea cuyo número es **"número"**

**GRAPHICS PAPER ¡expresión!**

Fija el color de fondo de los gráficos, según el valor dado por **expresión**
5 INPUT x
10 MODE 1: MASK 7: GRAPHICS PAPER x
20 MASK 255: MOVE 0,50: DRAW 640,50
40 GOTO 5

**GRAFICS PEN** expresión 1 expresión 2

Fija el color de tinta de los gráficos, según el valor dado en **expresión 1.** La **expresión 2** puede ser 0 ó 1. Cualquier de las dos expresiones puede ser omitida, pero no las dos.

10 INPUT x,y
20 GRAPHICS PEN x,y
30 DRAW 639,0

**HEX$ expresión 1 expresión 2**

Convierte la **expresión 1** en una cadena representativa de hexadecimal equivalente con tantos dígitos como indique **expresión 2.**
Si **expresión 2** no se da, o es demasiado pequeña, la cadena cubre los dígitos hexadecimales necesarios.
10 INPUT x, y
20 PRINT HEX$ (x,y)

**HIMEN**

Esta función guarda la dirección de memoria más alta disponible para el BASIC PRINT HIMEN
42619.



# PROGRAMAS COMENTADOS EN BASIC

## JUEGO DE BARCOS



El juego que proponemos es una versión simplificada del conocido juego de los barquitos.

La cuadrícula de juego se compone de 6 × 6 cuadritos, en la que se sitúan aleatoriamente 3 submarinos, (un solo cuadro cada uno). Pueden estar juntos o separados e incluso pegados a las pareces del cuadro.

El juego consiste en tratar de hundirlos en el mínimo número de jugadas. El ordenador nos irá indicando en cada jugada si las coordenadas del tiro (fila, columna), corresponden a "agua" o "hundido", en caso de disparar a un cuadro de forma repetida indicará "disparo ya realizado".

Al final del juego, una vez hundidos los tres submarinos, nos dará la calificación como jugador.

**COMENTARIOS AL PROGRAMA:**

**LINEAS**

10-40 Presentación y bucle de retardo (línea 40).

50 Dimensionado de una matriz de 6 × 6 (numérica), que emula la cuadrícula de juego, de manera que cada elemento de la matriz representa un cuadro del juego.

70-100 Situar barcos, las coordenadas de cada barco (fila y columna), se fian aleatoriamente mediante la línea 80, que genera dos números enteros comprendidos entre 1 y 6. Como al dimensionar la matriz, todos los elementos valen cero, a los seleccionados aleatoriamente se les da el valor 1, que representa una cuadrícula con submarino. Las que contienen un cero son agua.

La línea 90 comprueba que dos barcos no ocupen la misma casilla. El bucle termina al situar los 3 barcos (si se desean más variar el valor final del bucle).

120-150 Dibuja las líneas horizontales. Como el número de cuadros es 6, el de líneas será 7 (cuadros + 1). El paso del bucle es 16 ya que a cada cuadro en MODE 1 le corresponden 16 pixels o puntos elementales (alta resolución). El bucle se establece a partir de 16 para poder situar en la primera columna los números (1 al 6).

170-190 Este bucle es idéntico al anterior para dibujar las líneas verticales.

160 y 200-250 Imprimen los números de las columnas y de las filas, mediante la función LOCATE columna, fila, que fija el puntero de PRINT.

260-310 Entrada de las coordenadas del disparo y control de que éstas no sean mayores que 6. (Máximo número de filas y columnas). En caso de ser mayor vuelve a pedir la coordenada correspondiente (observar el cursor de INPUT en la pantalla).

320 Controla si la coordenada introducida lo fue ya anteriormente (tiro repetido). El valor 2 se fija cuando hay un disparo no repetido tanto si es agua como fuego, por la línea 380.

330 Control de "hundido", como a los barcos se les identificó en la matriz como 1, si el elemento de la matriz que corresponde a las coordenadas del disparo es uno, ahí se encuentra un barco. La variable BH contiene el número de barcos hundidos, a fin de controlar el fin de juego en la línea 370.

340 Control de "agua". Si el disparo no fue hundido ni repetido será "agua", representado en la matriz por el valor 0.

350 Contador de disparos.

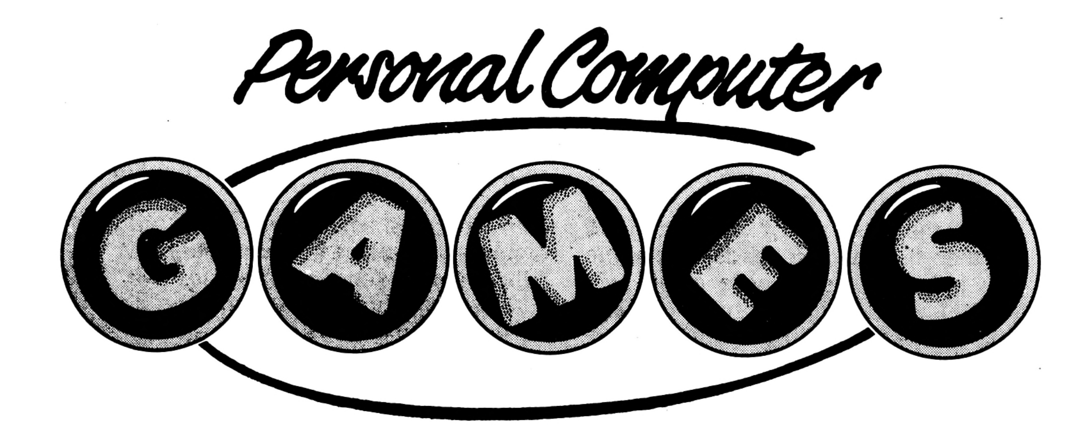
360 Borrado de textos viejos, para iniciar una nueva tirada.

370-390 Control fin del juego y apunte de tirada (2 en la matriz). Vuelta al juego (continuación).

410-440 Final del juego y establecimiento de la clasificación según el número de tiradas empleado en hundir los 3 submarinos. Se pasa a esta fase del juego por la línea 370, cuando BH = 3.

## MEJORAS AL PROGRAMA

El programa se ha construido pensando en su valor didáctico y no como un buen juego, por tanto el número de mejoras que pueden realizarse es grande, por ejemplo mayor cuadrícula, mayor número de barcos, barcos de más de una dimensión, mejor presentación, etc., algunas modificaciones serán fáciles de realizar, ¡anímese y hágalas! Otras serán más dificultosas, en cualquier caso inténtelo, es una buena forma de avanzar en programación.




## JUEGO DE LOS BARCOS
## VERSION SIMPLIFICADA

```
..............................
20 MODE 1:PRINT "JUEGO DE BARCOS. VERSIO
N SIMPLIFICADA"
30  PRINT :PRINT "SE CONSIDERA UNA CUADR
ICULA DE 6*6        Y TRES BARCOS DE UN SO
LO CUADRO"
40 FOR K=1 TO 10000: NEXT K
50 DIM C(6,6):  ' Dimensionamiento matriz
 de juego.
60  ' ................SITUAR BARCOS
```

```
70 FOR K=1 TO 3
80 FILA=INT(RND*6)+1 : COL=INT(RND*6)+1
90 IF C(FILA,COL)=1 THEN 80 ELSE C(FILA,
COL)=1
100 NEXT K
110 ' .........,........DIBUJO CUADRICULA

120 CLS
130 FOR Y=16 TO 16*7 STEP 16
140 PLOT 16,Y:DRAWR 16*6,0
150 NEXT Y
160 LOCATE 2,25: PRINT "123456"
170 FOR X=16 TO 16*7 STEP 16
180 PLOT X,16:DRAWR 0,16*6
190 NEXT X
200 LOCATE 1,19: PRINT "1"
210 LOCATE 1,20: PRINT "2"
220 LOCATE 1,21: PRINT "3"
230 LOCATE 1,22: PRINT "4"
240 LOCATE 1,23: PRINT "5"
250 LOCATE 1,24: PRINT "6"
260 ' .....................JUEGO

270  LOCATE 1,1: PRINT  "COORDENADAS DEL
 DISPARO"
280  LOCATE 1,3: INPUT "FILA ";FILA
290 IF FILA >6 THEN 260
300  LOCATE 1,4: INPUT "COL  ";COL
310 IF COL >6 THEN 260
320 IF C(FILA,COL)=2 THEN LOCATE 10,10:P
RINT "DISPARO YA REALIZADO":GOTO 260
330 IF C(FILA,COL)=1 THEN  LOCATE 15,4:
PRINT "HUNDIDO!": LOCATE COL+1,FILA+18:
PRINT CHR$(24);"*";CHR$(24):BH=BH+1
340 IF C(FILA,COL)=0 THEN LOCATE 15,4: P
RINT "AGUA    ":LOCATE COL+1,FILA+18: PR
INT CHR$(24);" ";CHR$(24)
350 disparos=disparos+1
360 LOCATE 10,10:PRINT "
   "
370 IF BH=3 THEN LOCATE 10,10 : PRINT "F
IN DEL JUEGO":GOTO 410
380 C(FILA,COL)=2
390  GOTO 260
400 ' ............. FIN DEL JUEGO

410 IF DISPAROS<25 THEN PRINT "ERES UN F
ENOMENO":GOTO 440
420 IF DISPAROS<30 THEN PRINT "NO ESTA N
ADA MAL":GOTO 440
430 PRINT "DEDICATE A OTRA COSA"
440 END
```

## II.3 CONCEPTO DE PROCEDIMIENTO

Podemos pensar que cuando el ordenador está en "toplevel", está esperando órdendes. Estas órdenes pueden ser instrucciones directas, según se vio antes, o bien puede ser la orden para la ejecución de un programa. Al comenzar la ejecución de un programa, el ordenador abandona el nivel superior, y entra a niveles inferiores. Este es otro de los posibles modos de operación. Se llama **MODO DE EJECUCION.**

El concepto de procedimiento es uno de los más ricos de todo el lenguaje LOGO y, desde luego, da origen a toda su estructura. Dicho brevemente, un procedimiento es un programa escrito en Logo.

En la definición de un procedimiento se pueden utilizar todas las instrucciones primitivas del lenguaje, así como los procedimientos que ya se hayan definido. Esto permite confeccionar los programas desglosando acciones largas y complejas en grupos de instrucciones que realicen una parte específica: los procedimientos constituyentes del anterior.

Por ello, un modo usual de trabajo en Logo, es escribir procedimientos útiles o utilidades que se pueden almacenar en memoria externa, como los discos. Cada vez que necesitemos alguno de ellos basta cargarlo en memoria y utilizarlo como un primitivo más.

Si el ordenador está ejecutando un procedimiento-1 que ha comenzado desde el nivel superior, decimos que está en nivel 1. A su vez este , podría "llamar" a otro procedimiento-2 que, de este modo, comenzaría a ejecutarse cuando el anterior, en realidad, todavía no había concluido. Decimos entonces que el Logo está en nivel 2. Y así sucesivamente. Este tipo de situaciones pueden representarse esquemáticamente como vemos en la figura II.1. En ella, cada línea horizontal representa un nivel; la más alta, el nivel superior, y cada paso hacia abajo representa la llamada a un nuevo procedimiento. Sólo cuando el procedimiento-2 termina por completo su ejecución, puede continuar el procedimiento-1 que le había llamado.

## II.4 DEFINICION DE PROCEDIMIENTO CON "TO" Y "END"

Hemos estudiado hasta aquí dos de los modos en que el Logo puede estar: "modo de órdenes directas" o "nivel superior", y "modo de ejecución". Otro modo en que puede estar es **MODO DEFINICION.** Se entra en este modo para definir un nuevo procedimiento.

Un procedimiento consta de tres partes: **NOMBRE, CUERPO** y **FINAL.** El nombre es una palabra que le identifica a todos los efectos. Para que una palabra pueda ser el nombre de un procedimiento, debe cumplir las siguientes condiciones:

— No puede ser el nombre de un primitivo.

— No puede coincidir con el nombre de un procedimiento ya definido.

— Su longitud ha de ser a lo sumo de 74 caracteres.

— Debe respetar todas las reglas de formación de las palabras.

El nombre de un procedimiento, en el AMSTRAD, puede estar formado con letras mayúsculas o minúsculas o una mezcla de ambas. Si dos procedimientos tienen el mismo nombre, pero uno en mayúsculas y otro en minúsculas, el ordenador los diferencia. La tecla CAPS LOCK, se utiliza como un interruptor para pasar de un tipo de letras a otro.

**EJEMPLO II.3**

— Nombres correctos de procedimientos:
a) dibujo
b) DIBUJO 2
c) Palabra.1

— Nombres incorrectos:
a) cs (porque es un primitivo)
b) pa/2 (porque el carácter "/" es un separador de palabras.
c) A B (porque incluye un espacio en blanco).

El cuerpo del procedimiento consite en todas las instrucciones que lo forman. Se escriben ordenadamente de arriba abajo y de izquierda a derecha. Entre cada dos instrucciones basta dejar un espacio en blanco, como entre cada dos palabras. Sin pulsar la teclar enter, se pueden escribir a lo más dos líneas.

Por último, el final del procedimiento indica sencillamente donde termina este.

Para situar a Logo en modo defínción, hemos de comenzar por decidir el nombre del procedimiento que vamos a definir. Supongamos que el nombre elegido es PALABRA. Entonces basta escribir desde el toplevel, to PALABRA
y pulsar enter después. Cuando el AMSTRAD entra en modo definición cambia el promopt para indicarlo: pasa "?" a ">". Todo lo que se escriba en modo definición lo tama el ordenador como constituyente del procedmiento que se está definiendo. Por consiguente no detecta ningún error; estos sólo se detectan en la ejecución. Tras cada instrucción pulsamos la tecla enter o bien dejamos un espacio en blanco antes de escribir la próxima.

Para que el ordenador abandone el modo definición, basta indicar que se ha llegado al final del procedimiento. Esto se hace escribiendo al comienzo de una nueva línea la palabra,

END

seguida de enter. Entonces Logo escribe en pantalla el mensaje,

PALABRA defined

que indica que el procedimiento llamado PALABRA forma parate ya de la memoria. El ordenador queda entonces en toplevel.



**EJEMPLO II.4**

— Si suponemos que el procedimiento llamado PALABRA está formado por las instrucciones INS1, INS2 e INS3, en la pantalla del ordenador veremos:

?to PALABRA
>INS1, INS2
>INS3
>end
?PALABRA defined

Obsérvese que se podía haber escrito INS2 en una línea distinta a la ocupada por INS1.

Otro modo de interrumpir la definición de un procedimiento es pulsando la tecla **ESC.** En este caso el procedimiento queda sin definir, cualquiera que sea el momento en que se haga. Esto obedece a un hecho general ya que:

"Al pulsar la tecla "esc", el Logo abandona lo que estuviera haciendo y regresa a toplevel, apareciendo en pantalla el mensaje **Stopped!**".

### II.5 EDICION DE PROCEDIMIENTOS

Finalmente, el Logo puede estar en **MODO EDICION.** Este modo puede ser usado sistemáticamente para definir procedimientos. Permite definir más de un procedimiento a la vez, y permite modificar los procedimientos ya definidos, y presentes en memoria. Se estudia por completo en el capítulo V.

# INTRODUCCION A LOS

## III.1 PRESENTACION DE LA TORTUGA

UNA de las cosas que cautivan desde el principio, cuando uno se aproxima al lenguaje Logo, es su capacidad para dar órdenes a una "supuesta" tortuga que le obedece ciegamente... si se le dan las órdenes correctamente. La tortuga, en efecto puede ser nuestra primera aproximación a lo que podría ser un pequeño **robot.**

Este robot "vive" en la pantalla de nuestro ordenador en la actualidad. Nada se opone a que de hecho sea el precursor de otras "tortugas" que sean capaces de moverse fuera de la pantalla: en nuestro mundo real. En realidad las tortugas actuales son sucesoras de otras que nacieron con el lenguaje, y que estaban conectadas por cables al ordenador. Estaban provistas de unas plumas de diversos colores y de goma de borrar, y hacían dibujos sobre el suelo obedeciendo las órdenes que recibían de su amo: claro está, le hablaba en lenguaje Logo.

Nosotros podremos hacer muchas cosas con la tortuga del Logo. Disponemos de muchas órdenes directas (los primitivos) y podemos inventar cuantas órdenes queramos. La forma que tiene la tortuga en el AMSTRAD recuerda a la de una pequeña flecha. Pero lo mejor es que la conozcamos ya:

Carguemos el Dr. LOGO en el AMSTRAD y escribamos nuestra primera orden a la tortuga:

(en minúsculas y seguido de "enter").

Esta es la orden para decir a la tortuga que aparezca en pantalla, en la posición en que esté situada. Entonces aparece la pantalla de gráficos y la tortuga situada en el centro... esperando nuestras órdenes.

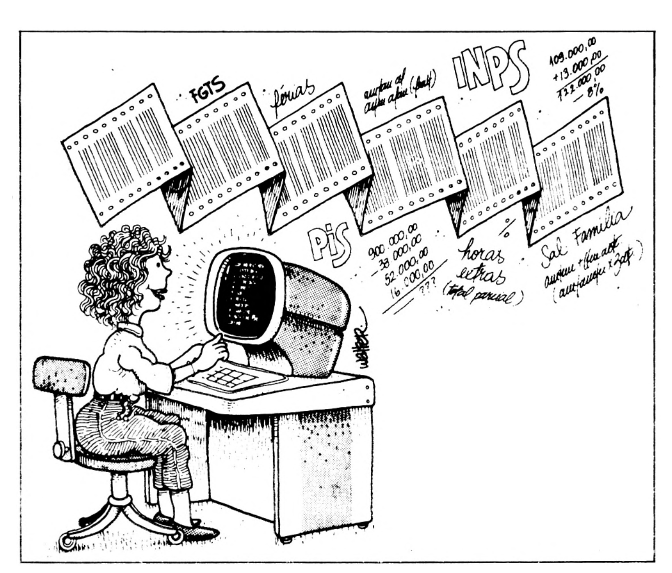
Podemos hacer invisible a la tortuga, sin que por ello abandone su actual posición en la pantalla ni ninguna de sus otras características. Simplemente dejaremos de verla pero ella estará ahí. Esta orden es:

**ht**

Pongamos atención sobre un hecho ya mencionado: cada orden que se refiera a una característica de la tortuga, cambia precisamente esa característica y respeta el estado de las demás.

## III.2 SOBRE LOS MENSAJES DE ERROR

SI hubiéramos escrito la orden anterior incorrectamente, la tortuga no nos hubiera entendido, como es natural. Probemos, por ejemplo a darle la orden anterior pero escribiendo cada letra en mayúsculas. Inmediatamente después de que pulsemos la tecla enter aparece el siguiente mensaje en la pantalla: 'I don't know how to




ST', ('No sé cómo se hace ST'). Es un mensaje de error.

Siempre que demos una orden (una instrucción) incorrectamente, aparecerá algún mensaje de error. El ordenador no se estropeará por eso, ni la tortuga... ni nadie. Lo que sí debemos hacer cuando veamos un mensaje de error es pararnos a analizar qué es lo que ha pasado, qué es lo que hemos hecho equivocadamente para que podamos poner remedio y sobre todo, para que no nos vuelva a suceder. Esta es la forma más razonable de aprender.

Si el error se produce cuando estamos trabajando en modo directo bastará con que escribamos de nuevo la orden pero ahora correctamente. Por el contrario si un programa se para cuando todavía no había terminado y emite un mensaje de error entonces deberemos reescribir la instrucción que lo ha provocado, pero ahora dentro del programa. Ya aprenderemos cómo se hace esto.

Hay muchos mensajes de error que podemos recibir de parte del Logo. Intentan dar una indicación de la razón del error. A lo largo de estos capítulos procuraremos ir indicando los tipos de errores más frecuentes asociados con lo que estemos viendo en cada momento. Tomemos nota entonces de que si una orden no la reconoce la tortuga (el Logo en general), recibiremos el mensaje anterior ('No sé cómo se hace...').

## III.3 ESTADO DE LA PANTALLA

NO sólo se puede trabajar con la tortuga en Logo. También se pueden escribir programas en los que apenas aparezca o en los que no aparezca en absoluto. Incluso se pueden hacer algunos gráficos sin usar la tortuga.

Para los programas que no usen gráficos el lenguaje Logo dispone de una pantalla dedicada por completo a texto. En ella no se puede hacer ningún gráfico. La llamaremos pantalla de texto. Para situarla se usa la orden

# GRAFICOS TORTUGA



**ts**

Es una abreviatura de la palabra textscreen que en inglés significa pantalla de texto. Hagamos la prueba en el ordenador y veremos cómo perdemos de vista a la tortuga. Vemos que aparece el prompt **?** y a su derecha el cursor (que es un cuadrado).

Cada vez que demos una orden que afecte a la tortuga se instalará la pantalla de los gráficos, para que la tortuga nos pueda obedecer. Pero existen dos pantallas de gráficos: una mixta gráficos-texto en la que se reservan algunas líneas para texto en la parte inferior y otra de sólo gráficos. Más adelante veremos que, incluso en el terreno reservado a la tortuga, es posible escribir texto.

La orden que instala la pantalla de sólo gráficos es:

**fs**

Es abreviatura de fullscreen. Cualquier orden que afecte a la tortuga instala la pantalla mixta gráficos-texto. Pero además nosotros podemos instalarla cuando lo creamos más conveniente sin más que escribir la orden:

**ss**

que es abreviatura de splitscreen.

Cuando trabajamos en Logo, hemos de tener en cuenta siempre que las cosas son relativas; que cada momento sucede a otro y que recoge siempre las condiciones que el precedente le transmite. Así por ejemplo supongamos que ejecutamos un procedimiento que, al acabar, deja la pantalla en modo mixto. Si después ejecutamos otro en el que se escriban muchas cosas en pantalla, y queremos que se vean, es necesario pasar a la pantalla de sólo texto: de otro modo sólo se verían las líneas inferiores de la pantalla. Esto no hubiera pasado si el primero que ejecutamos hubiera sido el segundo de los programas. En realidad basta con poner la orden que sitúa la pantalla de texto al principio de cada programa que la necesite. Aunque ahora no nos demos cuenta del todo, este hecho es de la mayor importancia.

## III.4 MOVIMIENTO RELATIVO Y MOVIMIENTO ABSOLUTO

Enseguida estaremos dando órdenes a la tortuga para que haga dibujos en la pantalla. Pero antes es preciso que hablemos un poco de la pantalla de los gráficos que es el mundo de la tortuga.

La tortuga se mueve por la pantalla y podemos pensar que para moverse va dando pasos. En realidad si hacemos que camine un número de pasos suficientemente grande entonces sobrepasará los límites de la pantalla y... dejaremos de verla. Pero ella seguirá obedeciendo nuestras órdenes aunque no la veamos. Puede hacer dibujos muy grandes de los que sólo veamos una pequeña parte.

¿Cuántos pasos puede dar la tortuga horizontal y verticalmente, de modo que permanezca dentro de los límites de la pantalla? Horizontalmente la pantalla mide 640 pasos y verticalmente 400. De modo que si la tortuga está situada en el centro de la pantalla puede dar 320 hacia la derecha o la izquierda y 200 hacia arriba o abajo sin salirse de la pantalla. Atención porque cuando esté situada la pantalla mixta no podremos ver a la tortuga si la situamos en la zona de texto. Dentro de un momento haremos las primeras pruebas.

Existen dos clases de órdenes que hacen moverse a la tortuga. Unas, las más usadas, indican un movimiento relativo: el resultado (la nueva posición de la tortuga) depende de la situación que tenía cuando se le dio la orden. De esta manera, la misma orden dada en momentos diferentes dejará a la tortuga situada en lugares distintos. Por ejemplo es el caso de que le ordenemos algo así como: 've para adelante 10 pasos'. Es lo que se llama 'movimiento relativo' de la tortuga.

Además existe el 'movimiento absoluto'. Podemos imaginar que tenemos asociado con cada punto de la pantalla un par de números: el número de pasos que debe dar la tortuga horizontalmente y verticalmente para situarse en él, a partir del centro de la pantalla. Son las coordenadas de los puntos. Así, las coordenadas del centro son [0 0] porque este punto es el origen. Un punto que esté situado en la misma vertical que el origen pero por ejemplo 100 pasos de tortuga hacia arriba será el [0 100]: horizontalmente no ha de avanzar la tortuga para ir a él, pero verticalmente ha de dar 100 pasos.

También podemos dar a la tortuga órdenes para que se sitúe en el punto que nosotros queramos, sabiendo las coordenadas. Estas órdenes corresponden al movimiento absoluto, porque el resultado es inde-

pendiente de la situación que tenía la tortuga cuando la recibió. Tomemos nota de que hemos representado los puntos en coordenadas por dos números encerrados entre **corchetes** y separados por un espacio en blanco. En el primer capítulo decíamos lo que es una palabra y lo que es una lista. Como vemos, cada uno de los números es una palabra y hemos representado el punto por una lista [N M] en la que el primer elemento es la coordenada horizontal y el segundo la vertical.

### III.5 PRIMEROS PASOS DE LA TORTUGA

Ahora estamos en condiciones de dar órdenes a la tortuga para comenzar a hacer dibujos. Comenzaremos estudiando el movimiento relativo y más adelante veremos el absoluto. Sería deseable poder ordenar a la tortuga que vaya hacia adelante o hacia atras, o que gire a derecha o izquierda, etc. Todas estas órdenes y muchas otras se le pueden dar.

En cada momento la tortuga "mira" hacia alguna dirección. Cuando le ordenemos que avance hacia delante se moverá en esa dirección. La orden que hace que avance N pasos es:

**fd <N>**

en la que N es un número que **tenemos** que escribir dejando un espacio en blanco después de la palabra fd (por eso lo hemos escrito encerrado entre <y>. Entonces la tortuga avanza N pasos y deja una línea dibujada por donde va pasando. Si se nos olvida escribir el número, entonces aparecerá en pantalla el mensaje,

**Not enough inputs to fd**

que significa 'no suficientes datos para fd'. Es un mensaje de error que nos aparecerá siempre que se nos olviden datos. La palabra fd es abreviatura de forward que, en inglés, significa adelante.

Para que la tortuga se mueva hacia atrás la orden es:

**bk < >**

donde todos los detalles son como antes (incluso el posible mensaje de error). La palabra bk es abreviatura de back (atrás).

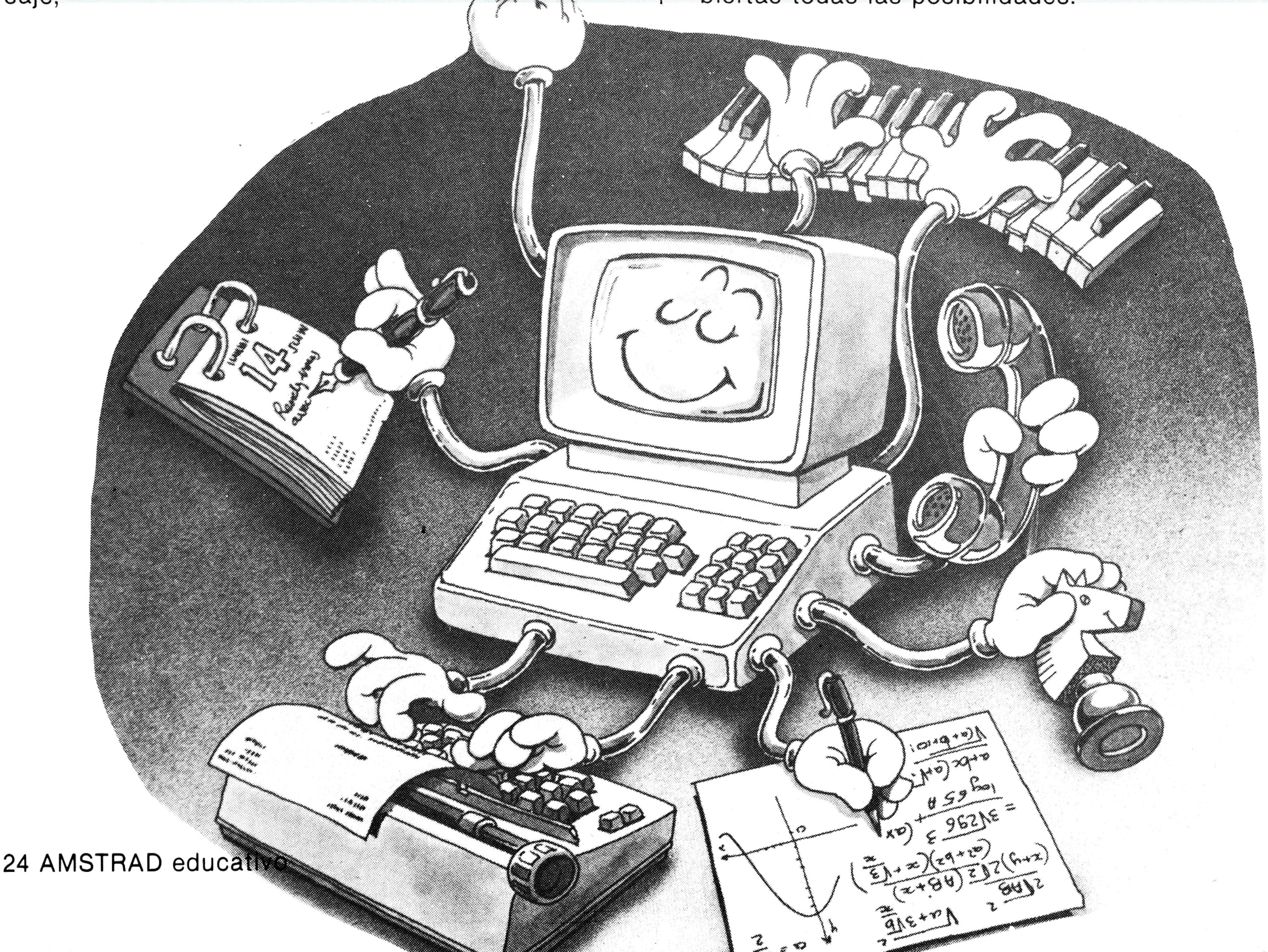
Claro que sin poder hacer que la tortuga gire, apenas podemos hacer dibujos de interés. Podemos hacer que la tortuga gire hacia la derecha o la izquierda con las siguientes órdenes:

**rt <a>**

para girar hacia la derecha (de la tortuga, ¡no la nuestra!), y

**lt <a>**

para girarla hacia su izquierda. En ambos casos la letra a representa el número de grados que queremos que gire. Es capaz de girar incluso un grado. Para ángulos pequeños la posición de la tortuga parecerá parecerá no cambiar, pero el giro lo habrá hecho. Con giros comprendidos entre 0 grados y 360, tenemos cubiertas todas las posibilidades.



Con estas sencillas instrucciones ya podemos hacer muchos y variados gráficos tortuga. Tenemos que tener siempre en cuenta un hecho importante:

Como el ángulo llano mide 180 grados, cuando queramos que la tortuga dibuje un ángulo de medida a grados, debemos girarla 180-a.

### EJEMPLO III.1

Vamos a hacer que la tortuga dibuje un ángulo de 45 grados, formado por dos lados de longitudes 50 pasos. Las órdenes serán:

```
ft 50
rt 135
fd 50
```

Esto es así porque hemos calculado que 180 - 45 = 135. ¿Podría haberlo calculado la tortuga? Sí. Las órdenes que hemos estudiado permiten que en vez de un número escribamos una operación entre números.

### EJEMPLO III.2

1) La siguiente secuencia de instrucciones produce el mismo resultado:

```
fd 50
rt 180-45
fd 50
```

2) Es muy sencillo dibujar un cuadrado. Basta hacer avanzar a la tortuga la cantidad que queramos que mida cada uno de los lados, y girar cada vez 90 grados. Hagámoslo:

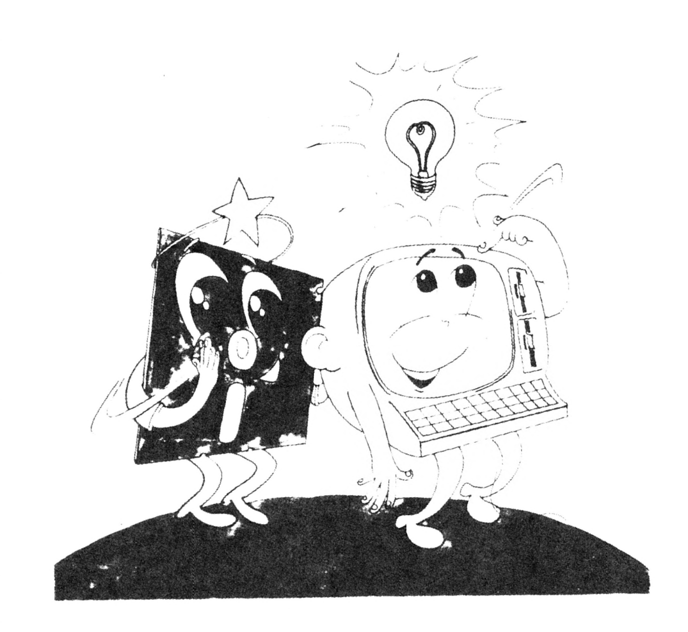
```
fd 100 lt 90
fd 100 lt 90
fd 100 lt 90
fd 100 lt 90
```

En este último ejemplo hemos escrito más de una orden en la misma línea. Ya sabemos que esto se puede hacer, siempre que incluyamos un espacio en blanco entre cada instrucción y la siguiente. Además hemos hecho el cuadrado girando hacia la izquierda. Por último vale la pena observar que el último giro que hemos hecho no es necesario para dibujar el cuadrado. Pero haciéndolo hemos conseguido que la tortuga acabe de dibujar el cuadrado en la misma posición que tenía cuando lo empezó. Ya veremos que esto es muy importante para algunas cosas.

### EJERCICIO III.1

1) Escribir las órdenes necesarias para que la tortuga dibuje un cuadrado de lado 200 girando hacia la derecha.

2) Dibujar un rectángulo cuyos lados midan 100 y 200 pasos de tortuga.



3) Dibujar los dos lados de un ángulo de 30 grados (cada lado mide lo que se quiera). (Atención: ¡no ha de girar 30 grados!).

4) Dibujar los dos lados de un ángulo de 60 grados.

## III.6 BORRADO DE LOS GRAFICOS

A estas alturas, si se han seguido en el ordenador los ejemplos y los ejercicios, la pantalla estará un poco 'sucia'. Es necesario aprender cuanto antes a borrar los gráficos.

Una primera forma es saliendo de la pantalla de gráficos. Así si escribimos ts pasaremos a la pantalla de texto. Si después regresamos a los gráficos con ss o con fs, la tortuga seguirá donde estaba, pero se havemos a la tortuga:

Podemos además borrar los gráficos sin salir de la pantalla del siguiente modo: la instrucción

**cs**

borra los gráficos que haya, sitúa a la tortuga en el centro de la pantalla (de coordenadas [0 0]) y mirando hacia arriba, e instala la pantalla mixta si se ejecuta desde la pantalla de texto o desde la mixta. Es pues una instrucción muy conveniente cuando queremos recomenzar a trabajar en un gráfico.

Otras veces sin embargo estaremos interesados en borrar los gráficos sin afectar a la situación de la tortuga. Esto se hace con la orden,

**clean**

Cuando se ejecuta, se borran los gráficos y se instala la pantalla mixta si se estaba en la de texto (todas las órdenes que afectan a la tortuga hacen esto último).

**EJERCICIO III.2**

1) Dibujar el perfil de los peldaños de una escalera.

2) Escribir las órdenes necesarias para borrar la pantalla y dibujar un triángulo equilátero de lado 100, desde el centro de la pantalla y girando la tortuga hacia la derecha. Repítase luego, girando hacia la izquierda y con el lado igual a 230. (Nota: recuérdese que los tres ángulos de un triángulo equilátero miden 60 grados).

3) Dibujar dos triángulos equiláteros que tengan un lado común.

## III.6 DIBUJO DE POLIGONOS REGULARES

Un polígono regular está formado por el mismo número de lados que de ángulos. De modo que si la tortuga recorre uno cualquiera de ellos, cuando llegue al punto de partida habrá girado precisamente 360 grados (una vuelta completa). Como todos los ángulos son iguales resulta que si el polígono tiene N lados y N ángulos cada vez la tortuga habrá tenido que girar 360/N.

Esto nos permite dibujar todos los polígonos regulares.

**EJEMPLO III.3**

1) Pentágono regular de lado 120, de modo que no vemos a la tortuga):

```
cs
ht
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
fd 120 rt 360/5
```

2) Hexágono regular de lado 90:

```
clean
st
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
fd 90 lt 360/6
```

3) Dos cuadrados iguales que tienen un lado común:

```
cs
ht
bk 50 rt 90
bk 50 rt 90
bk 50 rt 90
bk 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
```

**EJERCICIO III.3**

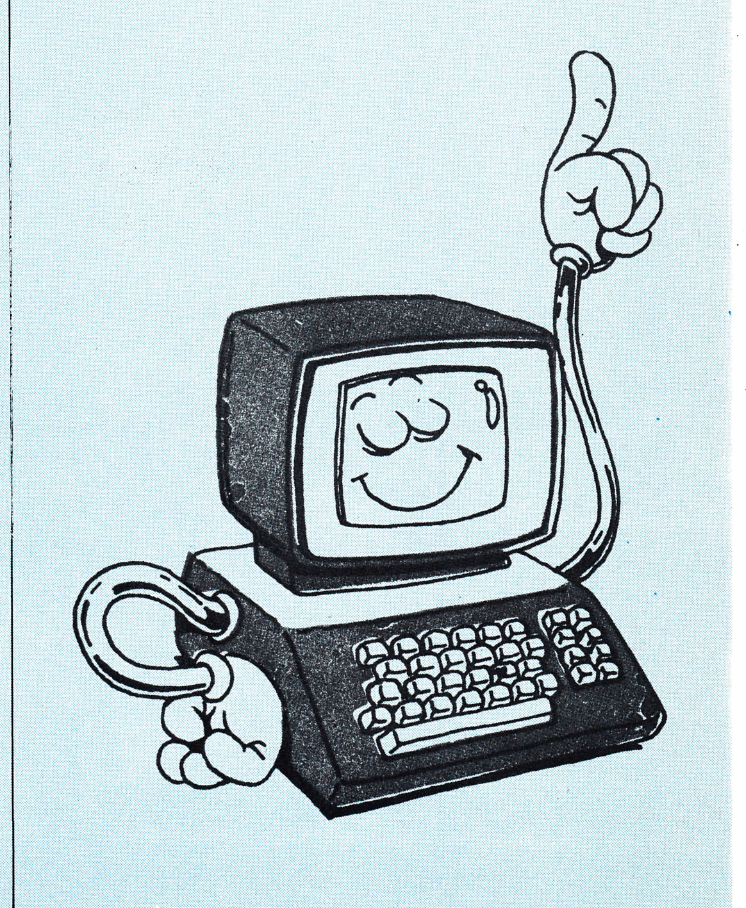
1) Dibujar un octógono (8 lados) regular de lado 70 y, sin borrarlo, otro desde donde esté la tortuga al final del primero pero de lado 50.

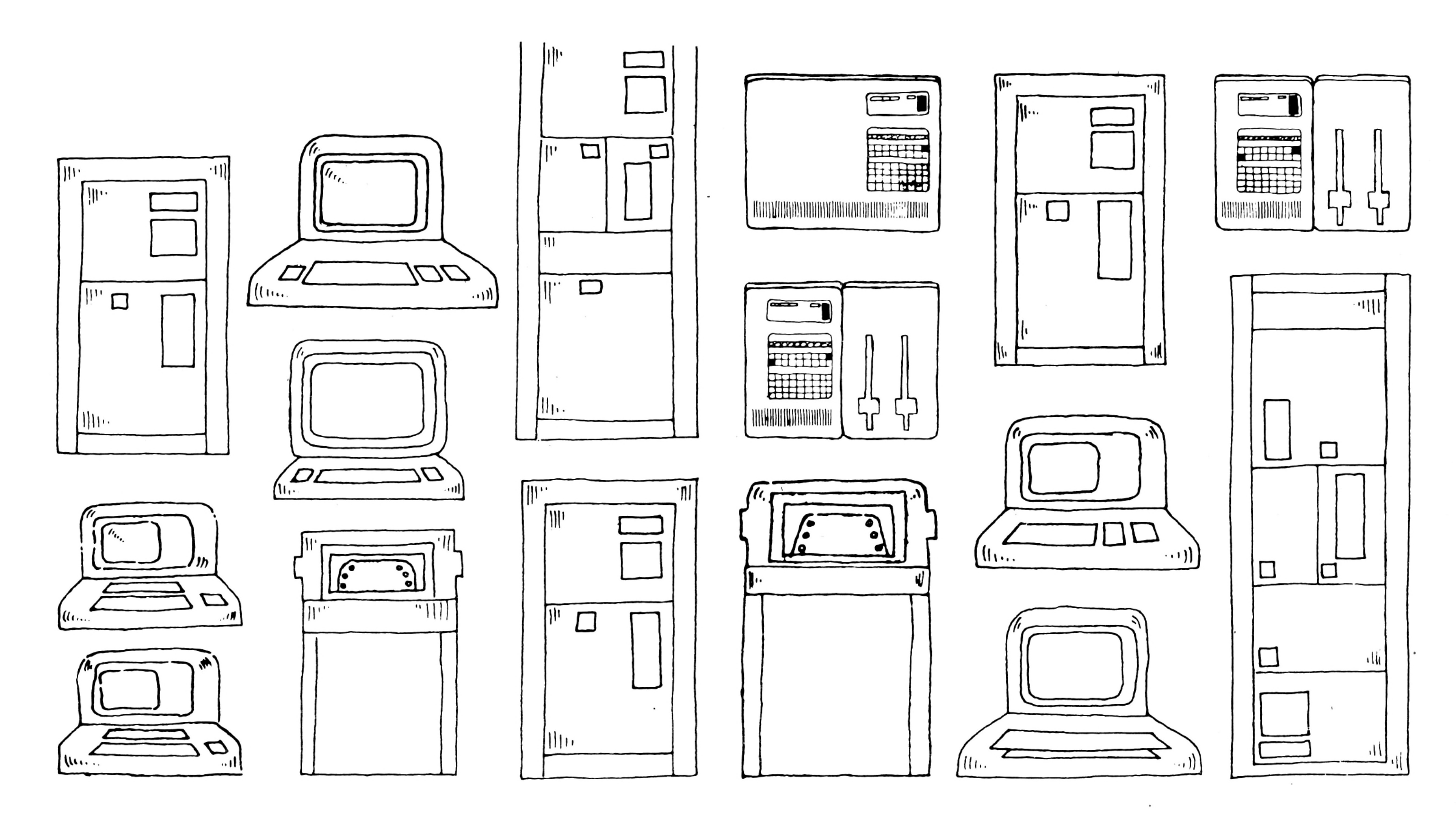
2) Dibujar 3 eneágonos (9 lados) encajados (como los octógonos anteriores).

## III.8 EDICION DE UNA LINEA DIRECTA

Por mucho cuidado que hayamos puesto al hacer en la máquina los ejemplos y ejercicios, es inevitable que nos hayamos equivocado más de una vez mientras estamos escribiendo las órdenes en el teclado. Si pulsamos la tecla "enter" y la orden contiene un error, no se puede ya corregir. Recibiremos, en tal caso, el correspondiente mensaje de error y a continuación podemos proceder a escribirla de nuevo.

Si advertimos el error antes de haber pulsado enter, entonces podemos rectificarlo sin tener que reescribir toda la línea. Las acciones que permiten corregir una línea se llaman acciones de edición de esta. Con las teclas de movimiento del cursor a la izquierda y derecha (→ y ←) podemos situar el cursor en el lugar necesario. Si escribimos encima de otros caracteres, estos no serán sustituidos; por el contrario lo que escribamos se insertará entre los caracteres que queden a la izquierda del cursor y el caracter sobre el que está situado este. Para borrar el caracter situado





a la izquierda del cursor se utiliza la tecla rotulada 'DEL'. Para borrar el caracter situado bajo el cursor se utiliza la tecla rotulada 'CLR'.

Como máximo, una línea de instrucciones (línea lógica) puede ocupar 2 renglones (2 líneas físicas) antes de pulsar enter. Si tenemos una línea de instrucciones que ocupe dos renglones, podemos pasar de uno a otro con las teclas de movimiento vertical del cursor.

## APENDICE: SOBRE LA INSTRUCCION "REPEAT"

Aunque desde el punto de vista puramente estructural tal vez no sea este el lugar más adecuado, vamos a hablar en este apéndice al capítulo III de una instrucción que nos puede ahorrar muchos minutos de escritura en el teclado. Esto independientemente de que más adelante volvamos sobre ella.

En programación es muy importante la creación de rutinas (algoritmos) que repetidos un número de veces por el ordenador, produzcan el resultado deseado.

Así en nuestro EJEMPLO III.3.1 y III.3.2., hemos escrito repetidamente la misma secuencia de dos instrucciones. El EJERCICIO III.3.1 y III.3.2 ha puesto al lector en la necesidad de escribir muchas veces la misma cosa.

Existe en Logo una instrucción que viene a resolver el problema, cuando el número de veces que ha de repetirse un grupo de instrucciones depende de un valor numérico. La forma de la instrucción es:

**repeat** <N> <[lista de instrucciones]>.

La palabra 'repeat' ha de escribirse en minúsculas. El número N (es obligatorio poner un número o una operación entre números) indica cuántas veces ha de repetirse la lista de instrucciones. La lista de instrucciones es la secuencia de instrucciones que queremos que se ejecute N veces. Cada una de las instrucciones ha de separarse de la que le precede y de la que la sigue por uno o más espacios en blanco. No es necesario poner ninguna instrucción dentro de los corchetes, pero estos son indispensables. Se pueden incluir tantas órdenes como quepan en las dos líneas.

Usando esta instrucción algunos de los ejemplos ya vistos son mucho más cómodos y manejables.

## EJEMPLO III.4

1) (Como el ejemplo III.3.1)
```
cs ht
repeat 5 [fd 120 rt 360/5]
```
2) (Como el ejemplo III.3.2)
```
clean st
repeat 6 [fd 90 lt 360/6]
```
3) (Como el ejemplo III.3.3)
```
cs ht
repeat 4[bk 50 rt 90]
repeat 4[fd 50 lt 90]
```

(Nota: no ha de dejarse obligatoriamente espacio entre el número N y [ o entre [ y el caracter siguiente porque los caracteres '[' y ']' son separadores de palabras. Pero pueden dejarse).

# EL NUMERO OCULTO

**E**l presente programa tiene como objeto el jugar al número oculto con la máquina. El ordenador piensa un número de 0 a 99 y se dispone de 6 intentos para averiguarlo. Para ejecutarlo basta escribir "numoc" después de escribirlo en el teclado.




LISTADO

```
to numocu
ct                                              BORRA PANTALLA.
make "oculto random 100                         GENERA EL NUMERO
pr [EL JUEGO DEL NUMERO OCULTO.]                TITULOS
pr [- - - - - -.- - - - - - - -]
pr [] pr[] pr[]
pr [TIENES QUE AVERIGUAR EL NUMERO OCULTO.]
pr [TIENES SOLO 6 OPORTUNIDADES.]
pr [CUANDO ESTES PREPARADO PULSA ENTER.]
make "listo rq                                  ESPERA A QUE SE PULSE ENTER
ct
make "numin 0                                   NUM. INTENTOS=0.
repeat 6[intento if :numero=:oculto [stop]]     REPITE 6 VECES EL PROCEDI-
                                                MIENTO intento Y CADA VEZ
                                                ANALIZA SI SE AVERIGUO EL
                                                NUMERO.
end

to intento
type [CUAL ES EL NUMERO?]                       NO HACE RETORNO DE CARRO.
```

```
make "numero rq                                         PIDE EL NUMERO AL TECLADO
                                                        Y LO ALMACENA EN LA VARIA-
                                                        BLE LLAMADA numero.
if :numero=:oculto [pr[LO HAS CONSEGUIDO                SI SE HA ACERTADO EMITE
      EN  ] type :numin+1 type [ ] pr" INTENTOS]]       MENSAJE.
if :numero>:oculto [pr[EL OCULTO ES MENOR]              SI EL OCULTO ES MENOR INFOR-
     make "numin :numin+1]                              MA Y AUMENTA EL CONTADOR.
if :numero<:oculto [pr[EL OCULTO ES MAYOR]              COMO LA ANTERIOR.
     make "numin :numin+1]
end
```

2)   ** MUCHOS CUADRADOS **  (D)

Este programa realiza una figura que consiste en 10 cuadrados, cada uno de los cuales se obtiene girando el anterior 36 grados, para conseguir una figura regular muy interesante

```
to figura
cs fs                                                   BORRA LA PANTALLA E INSTALA
                                                        LA PANTALLA DE SOLO GRAFICOS
repeat 10 [cuadrado rt 36]                              REPITE 10 VECES EL PROCEDI-
                                                        MIENTO CUADRADO Y GIRAR 36
                                                        GRADOS A LA DERECHA.
end

to cuadrado
repeat 4[fd 100 rt 90]                                  POR 4 VECES AVANZA Y GIRA.
end
```

## EL PROGRAMA TECNICO DEL MES

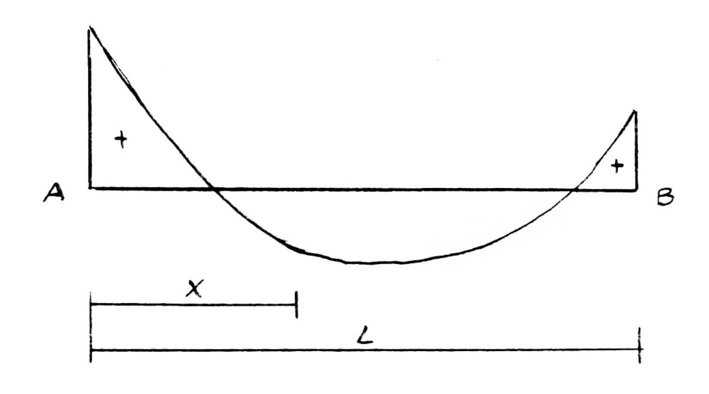
# LEYES DE MOMENTOS EN UNA VIGA

*Víctor J. Campo López*

UNA vez se ha resuelto por Cross u otro medio un pórtico o viga continua, queda la labor de calcular, viga por viga, las leyes de momentos obtenidos por el cálculo anterior, mas los debidos a las cargas sobre la viga, generalmente uniformes (incluido el peso propio).

Ya que la ley debida a una carga uniforme en una viga biapoyada es una parábola de segundo grado, construir la ley resultante, supone "colgar" dicha parábola desde los momentos debidos al Cross.

El momento resultante en un punto cualesquiera de la viga, será la suma algebraica de los obtenidos por tres efectos, el momento en A, el de B y el debido a la carga uniforme. Es decir,



$$Mx = MA\frac{L-X}{L} + MB\frac{X}{L} + Q^{X}\frac{(L-X)}{L}$$

Siendo: MA = Momento en A.
MB = Momento en B.
Q = Carga uniforme.
L = Longitud de la viga.
X = Punto considerado.

Existen tres puntos singulares A y B, que son conocidos (extremos) y el punto en el que el momento negativo es máximo.

El programa calcula el momento máximo central (no en el centro necesariamente), y los momentos a lo largo de la barra, en varios puntos, lo que permite ver la evolución de los mismos y los cambios de signo (puntos de corte con la barra, de momento = 0).

Además el programa dibuja gráficamente la ley de momentos resultantes.

Por último y para el caso de hormigón armado, calcula las capacidades mecánicas de la armadura, para los momentos máximos. En el caso en que el canto elegido sea inferior al mínimo, el programa avisa de este hecho.

El dibujo utiliza en longitud, dos escalas distintas a fin de que no se salga de la pantalla. Las escalas verticales y horizontales son distintas, para dar mayor claridad, pueden ser modificadas en la línea 330.

Las unidades utilizadas vienen indicadas en los propios INPUTS. Todas las longitudes se expresan en metros (m.), los momentos en metros por tonelada (m.T) y las cargas en Toneladas/metro (T/m), consecuentemente la capacidad mecánica se obtiene en Toneladas (T).

El criterio de signos puede modificarse, suprimiendo la línea 140.

Introduciendo los datos en el programa, en la forma:

— Longitud de la barra (m) = 6,0
— Carga uniforme (T/m) = 2,3
— Momento en A (m.T) = 0,875
— Momento en B (m.T) = 3,16
— Canto de la viga (m) = 0,4
— Ancho de la viga (m) = 0,3

Se obtienen en pantalla los siguientes resultados:

• Momento central máximo = 8,36 en X = 2,80 (el momento central es el máximo negativo según la figura, no tiene por qué coincidir con X = L/2, salvo en el caso que MA = MB).

• Tabla de valores de X, con los momentos correspondientes.

• Gráfico de la ley de momentos.

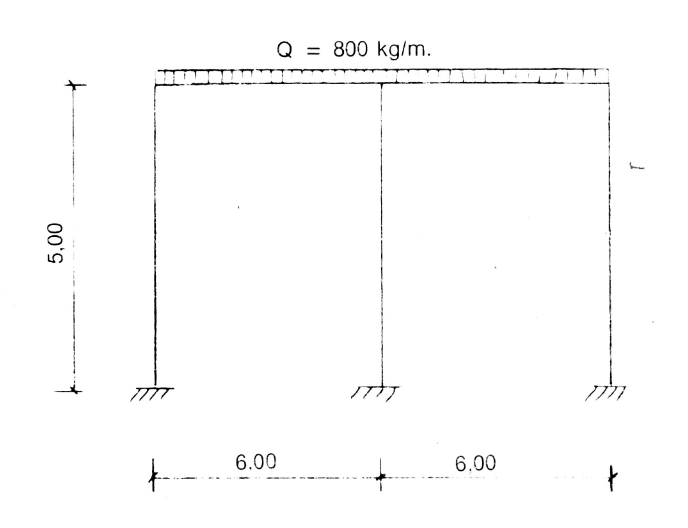
• Capacidades mecánicas a absorber por la armadura (hormigón armado).

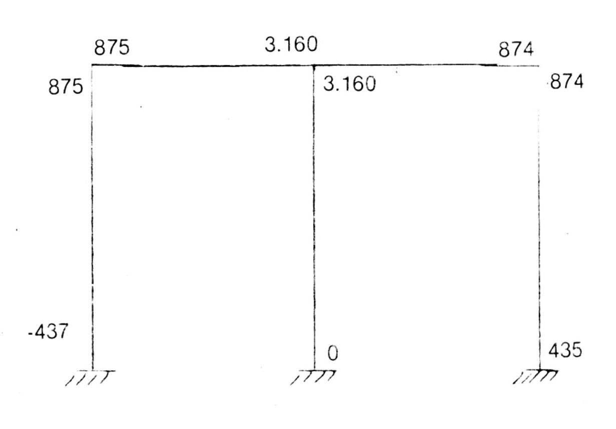
UA = 2,15 en el extremo A (empotramiento).
UB = 7,97 en el extremo B (empotramiento).
UC = 22,4 en el vano (corresponde al Mmax).

(El símbolo Pt equivale a #)

EJEMPLO

En el pórtico de la figura, se han obtenido por el método de Cross los siguientes momentos de empotramiento.



```
10 ' ..................................
15 '
20 '          LEY DE MOMENTOS
30 ' ..................................
35 '
41 MODE 1:  PRINT "   ESTE PGM   CALCULA
 LA LEY DE MOMENTOS EN UNA VIGA A PARTIR
 DE LOS MOMENTOS DE EMPOTRAMIENTO Y LA S
OBRECARGA UNIFORME (No se incluyen carga
s puntuales)."
42 PRINT : PRINT "   IMPRIME LA TABLA DE
MOMENTOS, GRAFICO Y CAPACIDADES MECANICA
S DE LAS ARMADURAS PARA LOS MTOS. MAXIMO
S (izquierdo, derecho y central)."
50 LOCATE 1,25: PRINT "PRESIONAR UNA TEC
LA PARA CONTINUAR"
60 Z$=INKEY$:IF Z$="" THEN 60
70 MODE 2
80  INPUT "Longitud de la barra (m)=",l:
 LOCATE 40,1
90  INPUT "Carga uniforme      (T/m)=",q
100  INPUT "Momento en A        (m.T)=",m
a: LOCATE 40,2
110  INPUT "Momento en B        (m.T)=",m
b
120  INPUT "Canto de la viga     (m) =",h
: LOCATE 40,3
130  INPUT "Ancho de la viga     (m) =",b
140 MA=-MA:MB=-MB
150 DEF FN A(X)=MA/L*(L-X)+MB*X/L+Q*X/2*
(L-X)
160 MMAX=-1000:XMAX=0
170 FOR x=0 TO 1 STEP 0.1
180 F=FN A(X)
190 IF F>MMAX THEN MMAX=F:XMAX=X
200 NEXT X
210 PRINT CHR$(24);"   Xmax="; USING "##
#.##    ";XMAX: LOCATE 40,4
220 PRINT "    Mmax="; USING "###.##    ";
FN A(XMAX): PRINT CHR$(24)
230 PRINT "  X      Mto.     X      Mto."
240 PRINT "------------------------------
-"
245 IF L<8 THEN S=0.5 ELSE S=0.75
```

```
250 FOR x=0 TO L STEP S
260 MX=FN A(X): PRINT  USING "₧₧₧.₧₧ ";X
;MX;
270 IF (X+S/2)>L THEN 300
280 MX=FN A(X+S/2): PRINT  USING "₧₧₧.₧₧
 ";X+S/2;MX
290 NEXT X
300 IF X<>L THEN PRINT  USING "₧₧₧.₧₧ ";
1;FN A(L)
310 GOSUB 450
320 ' DIBUJO
330 IF L<8 THEN EH=40:EV=10 ELSE EH=30:E
V=10
340 PLOT 250,200
350 DRAWR L*EH,0
360 PLOT 250,200:DRAWR 0,MA*-EV
370 PLOT 250+L*EH,200:DRAWR 0,MB*-EV
380  FOR X=0 TO L STEP 0.05
390 PLOT 250+X*EH,200-FN A(X)*EV
400 NEXT X
410 LOCATE 40,6: PRINT CHR$(24);" Ua=";
USING "₧₧₧.₧₧";UA
420 LOCATE 50,6: PRINT "  Uc="; USING "₧
₧₧.₧₧";Uc
430 LOCATE 60,6: PRINT "  Ub="; USING "₧
₧₧.₧₧";Ub: PRINT CHR$(24)
440 Z$=INKEY$:IF Z$="" THEN 440 ELSE RUN
 50
450 ' canto minimo y capacidades mecanic
as
460 m=ABS(ma):GOSUB 500:GOSUB 530:UA=U
470 m=ABS(mb):GOSUB 500:GOSUB 530:UB=U

480 m=ABS(mmax):GOSUB 500:GOSUB 530:UC=U

490 RETURN
500 hmin=1.69*SQR(m/b/1666)
510 IF h>hmin THEN RETURN
520 PRINT CHR$(24);"CANTO INFERIOR AL MI
NIMO";CHR$(24):RETURN
530 ' CAPACIDAD MECANICA
540 U=0.97*M/H*(1+M/B/H/H/1666)
550 RETURN
```

# NUESTRO PRÓXIMO NÚMERO




N.º 4

EL AMSTRAD y el CPM

Ficheros en el AMSTRAD

Basic del AMSTRAD

Fichas del AMSTRAD

Programando en Basic

Logo del AMSTRAD

Doctor Logo

El programa técnico del mes

# EDITORIAL

*Muchas gracias por el interés que estáis mostrando por nuestra revista, a través de vuestras cartas, opiniones y sugerencias.*

*Estamos ya tabulando estas indicaciones para irlas adaptando a nuestro esquema, y la planificación se irá ajustando, de esta forma, a nuestras necesidades.*

*Los programas irán gradualmente haciéndose más complicados y difíciles, a medida que vayáis dominando los niveles más elementales.*

*Si tenéis en mente algún programa que os gustase ver desarrollado, hacédnoslo saber y nos pondremos manos a la obra inmediatamente para que, cuanto antes, lo veáis publicado en vuestra revista.*

*El objetivo es que nuestro apelativo "Educativo" sea cada vez más eso, una plataforma que te permita dominar, cada día más a fondo, las posibilidades de tu aparato, formando un triángulo entre la revista, tu aparato y tú.*

*Esperamos vuestras opiniones.*



# SUMARIO

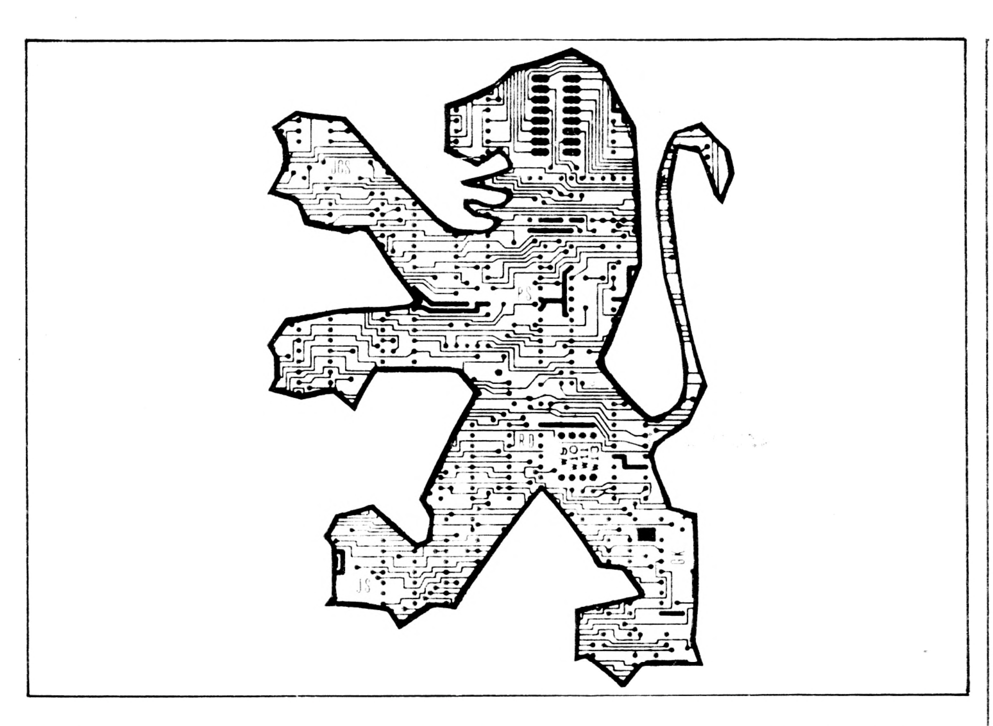



**Edita:** Grupo Editorial G.T.S., S. A.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02.
**Director:** Antonio Bellido.
**Colaboran** en este número:
Juan M. Pintor, Víctor J. Campo López.
**Maquetación:** Amalia Moreno.
**Secretaria de Redacción:** Mercedes Jamart.
**Publicidad:** Dpto. propio.
Bailén, 20-1.° Izda.
28005 Madrid
Telf.: 266 6601-02

**Fotocomposición:**
Fotocom, S. A.
**Fotomecánica:**
La Unión
**Imprime:**
Gráficas FUTURA Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23
Políg. Ind. Cobo Calleja
FUENLABRADA (Madrid)
**Distribuye:**
R.B.A. Promotora de Ediciones, S. A.
Travesera de Gracia, 56 Atico, 1.ª.
Tfno 93 200 82 56
**Depósito Legal:**
M. 8.904-1986

# EL AMSTRAD Y EL CP/M

**Por A. BELLIDO**



## EJERCICIOS CON UNIDADES DE DISCO

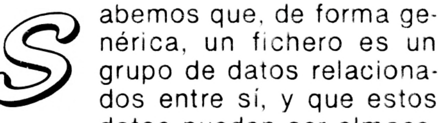
1.° ¿Cuál es la unidad de disco por defecto si la impronta en pantalla es "A>".
**Respuesta:** A.

2.° ¿Qué indica a CP/M la orden B?:
**Respuesta:** Que cambie la unidad de disco por defecto a la unidad.

3.° ¿Cuál será la impronta tras la orden anterior?
**Respuesta:** B>.

4.° ¿Es correcta esta orden?: Ñ:
**Respuesta:** No, porque la Ñ no es un caracter estándar ASCII y, además, y en todo caso, la versión 20 sólo permite en línea 16 unidades de disco.

5.° ¿Cómo interpreta esta orden?: A B: PRUEBA. UNO.
**Respuesta:** Siendo A la unidad por defecto, cargar en memoria el programa PRUEBA. UNO desde la unidad B.



## REFERENTE AL ALMACENAMIENTO EN DISCO DE UN FICHERO

Sabemos que, de forma genérica, un fichero es un grupo de datos relacionados entre sí, y que estos datos pueden ser almacenados en un disco. Centrándonos en este soporte, puede darse el caso de que un fichero no contenga ni un solo dato y, en el otro extremo, nada impide que los datos de un fichero ocupen toda la capacidad de un disco.

Estudiemos la forma en que se produce el almacenamiento de los datos de un fichero en un disco. Este es el tema que nos ocupará en el presente epígrafe.

En primer lugar, recordemos que la transferencia de información desde la RAM al disco, se hace secuencialmente, partiendo de la primera posición de memoria y hasta la última, pero en lotes de *bytes* equivalentes a un sector, regulado por un *buffer.*

Esto es cierto desde un punto de vista físico por las razones expuestas en su momento.

No obstante, con el fin de aumentar la eficacia de las transmisiones y la operatividad del sistema en su conjunto, la grabación de los *bytes* de información correspondientes a un fichero se realiza de acuerdo con ciertas normas.

Un fichero ocupará como mínimo un predeterminado espacio del disco denominado *tamaño del bloque,* que corresponderá a un número preciso de sectores.

El tamaño del bloque suele ser de ocho a dieciséis sectores. En lo que sigue, admitiremos que el tamaño del bloque es de ocho sectores, y que el sector es de 128 *bytes,* un fichero con menos de 1.024 *bytes* de información ocupará un bloque completo (8 sector < > 1.024 bytes) en el disco.

Ya conocemos el límite inferior de ocupación de disco por parte de un fichero. Pero, ¿qué sucede si un fichero requiere más de un bloque?

Sencillamente, irá ocupando bloques sucesivos, hasta completar la grabación en unidades completas del bloque. En nuestro supuesto, grupos de 1.024 bytes.

Es evidente que, para ficheros que requieran menos de un bloque, la capacidad de almacenamiento de un disco está subexplotada, pero, a cambio, el acceso a la información de cada fichero mejora indudablemente.

Por otra parte, para ficheros que requieran más de un bloque el aprovechamiento es aceptablemente alto, ya que sólo es dudoso el porcentaje de ocupación del último bloque.

Hagamos un alto y recordemos que cada fichero queda individualizado por su nombre, gracias al cual se le distingue de cualquier otro. También se hizo referencia a que cada fichero queda registrado, en la pista catálogo mediante un grupo de 32 *bytes* que reflejan todos los

datos necesarios que para el DOS pueda controlarlos.

Pues bien, ésto es cierto mientras el fichero no necesite más de un número predeterminado de bloques, para su almacenamiento. A este nú-nero de bloques se le conoce por *extensión.*

Dependiendo del ordenador, la extensión es, generalmente, de ocho a dieciséis bloques.

Por cada extensión que requiera un fichero se producirá una nueva entrada de 32 *bytes* en la pista catálogo para hacer referencia a la misma.

Por tanto, en base a todo lo expuesto, de suponer un fichero que ocupara la totalidad de un disco, diremos que el disco almacena un solo fichero, representado por su nombre, pero, en la pista catálogo aparecerán tantas entradas como extensiones (o grupos de bloques) permita la versión de CP/M implementada al ordenador:

Un ejemplo servirá para fijar las ideas.

Con los conceptos de sector, bloque y extensión en nuestro haber, analicemos un disco cuyas características resumidas, sean las que siguen:

— 40 pistas.
— 9 sectores por pista.
— 512 bytes por sector.
— 2 sectores por bloque.
— 16 bloques por extensión.
— 64 ficheros catalogables como máximo.

Es claro e inmediato, en función, de lo dicho, que nuestro disco permite manejar sectores de 512 *bytes,* el tamaño del bloque es de 1.024 *bytes* (2 sectores) y la extensión requiere 16.384 *bytes* (16 K<>1 bloque).

En cada cara se podrán almacenar 184.320 *bytes* (180 K), distribuidos en cuarenta pistas a razón de 9 sectores por pista o, lo que es igual, 180 K, grabados en 360 sectores.

El número máximo de ficheros permitidos es de 64, ya que la pista catálogo sólo admite esta cantidad, pero por cada extensión adicional que necesite un fichero disminuirá en una unidad este límite.

La ocupación mínima de disco requerida por un fichero, por pequeño que éste sea, es de 1.024 *bytes* (un bloque)

Si el disco contuviera las rutinas del DOS, éstas, quedarían almacenadas en las dos pistas exteriores, con lo cual la capacidad del disco para datos quedaría disminuida a 1.71 K. Este punto se verá con más detalle posteriormente.

## ESQUEMA DE ORDENES GENERALMENTE ACEPTADAS POR ORDENADORES CONTROLADOS POR CP/M

- **Ordenes**
  - **Permanentes**
    - User
    - Dir
    - Era
    - Ren
    - Save
    - Type
  - **Transitorias**
    - MOVCPM
    - SYSGEN
    - STAT
    - PIP
    - SUBMIT XSUB
    - ED
    - ASM
    - DDT
    - LOAD
    - DUMP
  - **Implementadas**
    - Format
    - Copy (Filecopy)
    - Diskcopy (Copydisk)
    - CHKDISK

## REFERENTE A LAS ORDENES PERMANENTES



En el esquema anterior se han resumido los distintos tipos de órdenes que CP/M puede aceptar: Permanentes, transitorias e implementadas.

Las órdenes permanentes *(built in commands)* también son denominadas "residentes", y lo son por pertenecer a un subconjunto de programas de CP/M situado en el CCP y consiguientemente, tras la carga del DOS en la memoria interna, están ubicadas en la RAM de acuerdo con lo expuesto en el epígrafe "UBICACION DE CP/M". En otras palabras, las órdenes permanentes están a disposición inmediata del operador.

Con objeto de fijar esta idea, y an-

ticipando parte de la explicación que exigen las órdenes transitorias, diremos que la diferencia operativa que existe entre éstas y aquéllas se deriva del hecho de que CP/M para ejecutar una orden de las llamadas transitorias necesita cargar del disco en la RAM el programa correspondiente a la orden en cuestión.

Dicho esto, concluiremos con un resumen de las órdenes permanentes que CP/M interpreta y que están situadas en el CCP:

USER
DIR
ERA
REN
SAVE
TYPE

A su estudio individualizado se dedican los siguientes epígrafes:

## USER

**Función asignada:** Asignar un **área de usuario** mediante un número comprendido entre 0 y 15, ambos inclusive, con lo cual todo el tratamiento y manejo de ficheros se referirá automáticamente a los situados en el área de usuario especificada, ignorando todos los asignados a diferentes áreas.

Esta orden está disponible en CP/M-86 y CP/M-80 versiones 2.0 y sucesivas.

**Forma de obtenerla:** Tecleando USER seguido de un número entre 0 y 15, ambos inclusive, representativo del área de usuario en el que se desea trabajar.

**Ejemplo:** Para trabajar, en el área de usuario número 2 daríamos la siguiente orden:

A >USER 2

Al pusar<cr >para hacerla efectiva, la pantalla simplemente nos devolverá la impronta de la unidad de disco por defecto (A >en este caso) indicando con ésto que la orden fue cumplida y el efecto subsecuente asumido.

**Comentarios:** Las consecuencias prácticas de esta orden se derivan del hecho de capacitar al operador a generar 16 distintos directorios dentro de la pista catálogo y de tal forma que aquellos ficheros que hayan sido creados y nominados en un disco estando activa un área de usuario determinado, simplemente no existirán para CP/M al cambiar de área de usuario.

Un área de usuario se mantiene en activo, hasta que una nueva orden USER lo cambia o se inicializa CP/M.

Al inicializar CP/M se sitúa en el área de usuario número 0.

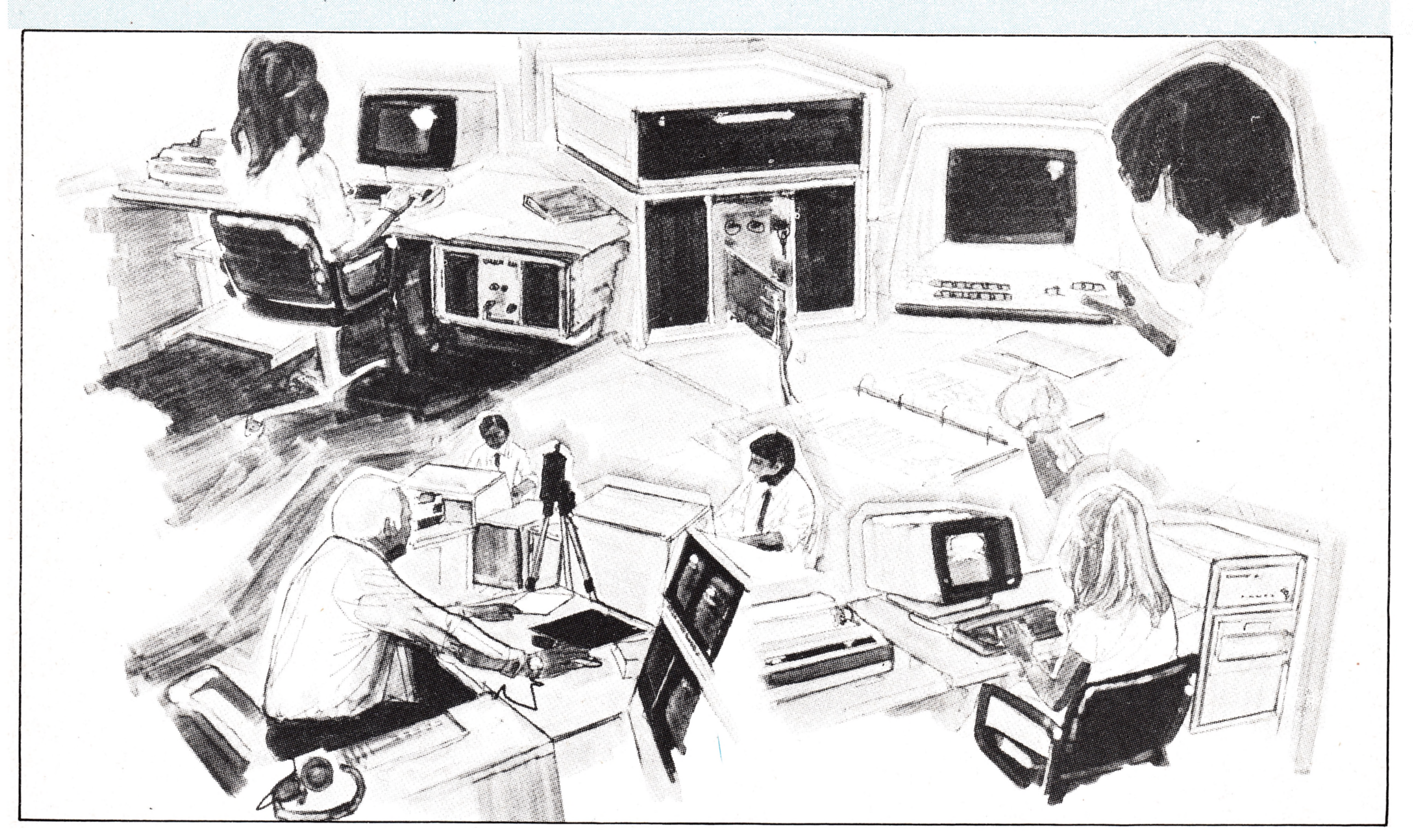
Al directorio de la versión 1.4 de CP/M-80 es compatible con el directorio del área de usuario 0 de la versión 2.0.

Los ficheros creados estando activo un determinado número de área de usuario, "existirán" sólo cuando esté activo este área de usuario. En otras palabras, al catalogar un nuevo fichero en la pista catálogo se le incluye y asocia el número de área de usuario en la que se trabaja en el momento de almacenarlo en el disco.

Por lo dicho, es fácil asumir que una orden del tipo DIR sólo mostrará el listado de los ficheros creados con el número de área de usuario que a la sazón esté activado.

Igualmente, un ERA ** borrará, exclusivamente, los ficheros generados con el número de área de usuario actual.

Por lo demás, toda referencia a ficheros se asume que es a los del área de usuario en la que se trabaja en ese momento.



## DIR

**Función asignada:** Mostrar una lista de todos o parte de los ficheros correspondientes al área de usuario actual contenidos en un disco situado en una unidad de disco específica. Dicho de otra forma, DIR retorna un directorio o catálogo de los ficheros referenciados del área de usuario activada.

**Forma de obtenerlo:** Con CP/M a nivel de operador teclear DIR seguido de la referencia a los ficheros —o fichero— de cuyos nombres se quiere obtener el listado, concluyendo la orden con<cr>

Si DIR no va seguido de referencia alguna a fichero, entonces el listado mostrará todos los nombres de ficheros contenidos en el área actual de usuario del disco en cuestión.

**Ejemplos:**

1.° Supongamos que se desea averiguar todos los ficheros contenidos en un determinado disco. Para ello, lo instalaremos en la unidad de disco por defecto —A, por ejemplo— y, en estas condiciones, damos la siguiente orden:

A > DIR

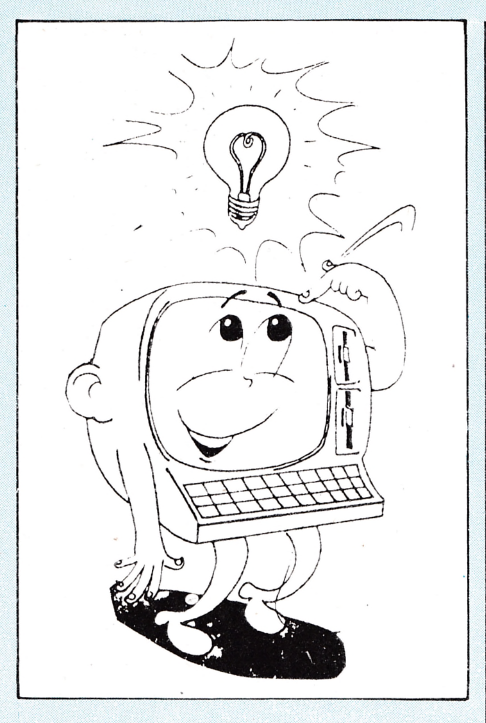
Al pusar<cr>para hacer efectiva la orden, veremos parpadear por unos instantes el piloto de la unidad de disco, y de inmediato aparecerá la lista completa de los nombres de los ficheros almacenados en el disco.

Por ejemplo:

```
A : PRUEBA REN : EJEMPLO SEI :
EJEMPLO SIE PROG SIE
A : LIBROS DOS : PROGLI DOS : LI-
BROS PRG : PRUEBA SAV
A : BASIC 1 : PROG DOS : LIBROS
BAK : LIBROS PRD
A : LIBROS SEC : BASIC 2 : BASIC 3
```

Para obrener el directorio de un área de usuario distinta a la presente, debemos comenzar por dar la orden USER n según se explicó en el epígrafe anterior, seguido de un DIR como el de este ejemplo.

En otro orden de cosas, y con la intención de completar la explicación dada en su momento respecto a P, veamos cómo se



puede obtener el listado anterior a través de la impresora, si el ordenador en uso lo permite. Con CP/M a nivel de operador, damos un P (CONTROL/P), con lo cual activamos la salida por impresora, y a continuación se ordena un DIR según lo anteriormente explicado. A partir de este momento irán imprimiéndose en pantalla y papel los nombres de los ficheros del disco.

Para desactivar la salida por impresora bastará pulsar CONTROL y P nuevamente.

2.° Si por cualquier razón deseamos obtener el directorio de nuestro disco en una unidad de disco —la B, por ejemplo— distinta a la que a la sazón sea la omitida, lo colocaríamos en la unidad elegida —la B— y, a continuación, daremos la siguiente orden:

```
A > B: DIR
```

Con lo cual conseguiremos nuestro actual objetivo.

3.° Si sobre el mismo disco y en la unidad A nos interesa averiguar exclusivamente cuántos ficheros son del tipo DOS, la orden será:

```
A > DIR * .DOS
```

Con lo cual, al ser ejecutada, tendríamos:

```
A: LIBROS .DOS: PROGLJ
.DOS: PROG .DOS
```

4.° Este nuevo ejemplo consiste en obtener todos los ficheros cuyo nombre es LIBROS, sea cual sea su tipo. Para ello, escribiremos:

```
A>DIR LIBROS.*
Siendo el listado subsecuente:
A : LIBROS .DOS: LIBROS
.PROG.: LIBROS .BAK: LIBROS
.PRD A: LIBROS .SEC
```

5.° Para concluir esta serie de ejercicios, utilizaremosa el comodín?, para localizar todos los ficheros cuyos nombres estén compuestos por cinco caracteres cualesquiera y sus tipos sólo tengan un caracter, sin precisar. La orden sería:

```
A > DIR??????
Y la respuesta:
A: BASIC.1: BASIC. 2: BASIC. 3
```

Observe que un DIR *.* es equivalente a un DIR, ya que en ambos casos nos estamos refiriendo a todos los ficheros.

Recuerde que todos los ejemplos anteriores son válidos para cualquier área de usuarios, con la única condición de dar la oportuna orden USER *n*, donde *n* es el número del área de usuario que se desea activar (ver USER).

**Comentarios:** Cuando CP/M nos muestra su impronta indicativa de que está a nivel operador (por ejemplo, A), ignora qué disco hemos instalado en una determinada unidad de disco; por esta razón, la primera acción que llevará a efecto tras recibir una orden que implique la manipulación del disco, será "tomar contacto" con él, con lo cual averiguará si efectivamente hay un disco colocado en la unidad indicada y, en tal caso, lo catalogará. A este proceso se le conoce en inglés por *log in.*

Esta operación la puede llevar a efecto el DOS gracias a la información contenida en la pista catálogo o directorio de la cual ya se dio razón en capítulo dedicado a "DISCOS", donde se advertía que al almacenar en un disco un nuevo fichero se grababa, en la pista catálogo una secuencia de treinta y dos *bytes.* Cada uno de estos **bytes** cumple la siguiente función:

El primero actúa como un semá-

foro que avisa al DOS que el fichero en cuestión puede ser leído. Una instrucción posterior de borrado de este fichero del disco haría cambiar el valor de este *byte,* con lo cual (CP/M consideraría que no existe.

Los ocho siguientes representarán los caracteres ASCII del nombre del fichero de izquierda a derecha. Si el nombre no requiere de los ocho bytes, los sobrantes a la derecha representarán códigos del espacio (32 decimal, 10 hexadecimal).

Los tres siguientes se refieren al tipo del fichero, en las mismas condiciones de los ocho anteriores.

El decimotercero indica el número de *extensión* actual dentro del fichero en cuestión.

A continuación, CP/M se reserva dos *bytes.*

El decimosexto contiene la medida de la *extensión* expresada en sectores.

Los restantes *bytes* indican la situación física dentro del disco de los diferentes bloques que ha requerido el fichero para ser almacenado.

## ERA

**Función asignada:** Borrar todos o parte de los ficheros contenidos en un disco instalado en la unidad de disco especificada. Los ficheros a borrar dependerán de la referencia que se haga a ellos y del área de usuario que esté activada.

**Forma de obtenerlo:** Con CP/M a nivel de operador teclear ERA seguido de la referencia al fichero —o ficheros— a borrar, concluyendo la orden con<cr> Si el área de usuario correspondiente a los ficheros interesados es distinta a la actual se debe dar, previamente la orden USER *n* oportuna (ver USER).

**Ejemplo:**

1.° Para borrar el fichero LIBROS .DOS de un disco instalado en la unidad de disco por defecto —la A, por ejemplo—, daremos las siguiente orden:

```
A> ERA LIBROS .DOS
```

Al pulsar<cr>para hacer efectiva la orden, y tras ser manipulado el disco, el fichero en cuestión habrá desaparecido del catálogo para CP/M. La operación se da por concluida al aparecer la impronta (A>) nuevamente. Con una orden DIR, según lo expuesto en el anterior epígrafe, se puede comprobar el nuevo directorio.

2 . °Para borrar todos los ficheros del disco, recurrimos al comodín "*" de esta forma:

```
A > ERA *.*
```

Con lo cual conseguiríamos el fin propuesto.

3 . °Para borrar todos los ficheros cuyo tipo sea DOS con una sola orden, escribiríamos:

```
A> ERA * DOS
```

Al pulsar <cr> se iniciaría la operación de borrado que concluirá con la aparición de la impronta (A>).

4 . °Para borrar todos los ficheros de nombre LIBROS, actuaríamos de forma similar:

```
A > ERA LIBROS.*
```

5 . °Para borrar todos los ficheros cuyo nombre empiece por PRO y el tipo por S, podríamos usar el comodín "?" así:

```
A> ERA PRO ?????.S??
```

6 . °Por último para borrar uno o varios ficheros de un disco situado en una unidad de disco distinta a la que actualmente está considerada por el DOS como unidad por defecto, utilizaremos el mismo criterio expuesto hasta el momento, pero anteponiendo la referencia oportuna a la unidad elegida. Por ejemplo:

```
A> ERA B: *.DOS
```

Con esta orden se borrarían todos los ficheros de tipo .DOS



del disco instalado en la unidad de disco B.

**Comentarios:** Con esta orden se borra sólo la información oportuna de la pista catálogo dedicada a los ficheros referenciados, con lo cual CP/M interpretará que el espacio ocupado por estos ficheros está disponible. Por el contrario, FORMAT, del cual ya se ha comentado algo, borra el disco entero y completamente.

## REN

**Función asignada:** Cambia el nombre de un fichero contenido en un disco instalado en la unidad de disco especificada, debiéndose tener en cuenta las siguientes normas:

1.° Está prohibido el uso de comodines, siendo necesario, por tanto, referirse al fichero por su nombre sin ambigüedad (ufn).

2.° El nombre "nuevo" no debe coincidir con ninguno de los existentes en el disco. Si fuera así, el DOS enviaría un mensaje FILE EXISTS (existe fichero) y la orden quedaría cancelada, apareciendo la impronta nuevamente.

3.° La orden REN sólo actúa sobre la unidad de disco especificada (la omitida o la explicitada). En ningún caso sobre dos unidades de disco distintas.

4.° El nombre "viejo" debe existir en el área de usuario actual del disco, de otra norma un mensaje del tipo NO FILE o NOT FOUND aparecería con lo que en definitiva nos viene a decir que no existe el fichero.

**Forma de obtenerla:** Con CP/M a nivel de operador, teclear REN, un espacio, seguido de la referencia, sin ambigüedad, del nuevo nombre a asignar al fichero, a continuación el símbolo "=" y el nombre actual del fichero.

**Ejemplos:**

1.° Supongamos que en el directorio de un disco instalado en la unidad A (admitida por defecto) existe un fichero denominado EJEMPLO SEI y pretendemos cambiarlo de nombre, siendo este EJEMPLO SIE. Si, con estas premisas, diéramos esta orden:

```
A> REN EJEMPLO SIE =
EJEMPLO SIE
```

Obtendríamos el mensaje FILE EXISTS, ya que hemos invertido el orden de los nombres: primero debe ir el nombre "nuevo" y después el "viejo".

2.° La orden correcta en el caso anterior es:

```
A> REM EJEMPLO SIE =
EJEMPLO SEI
```

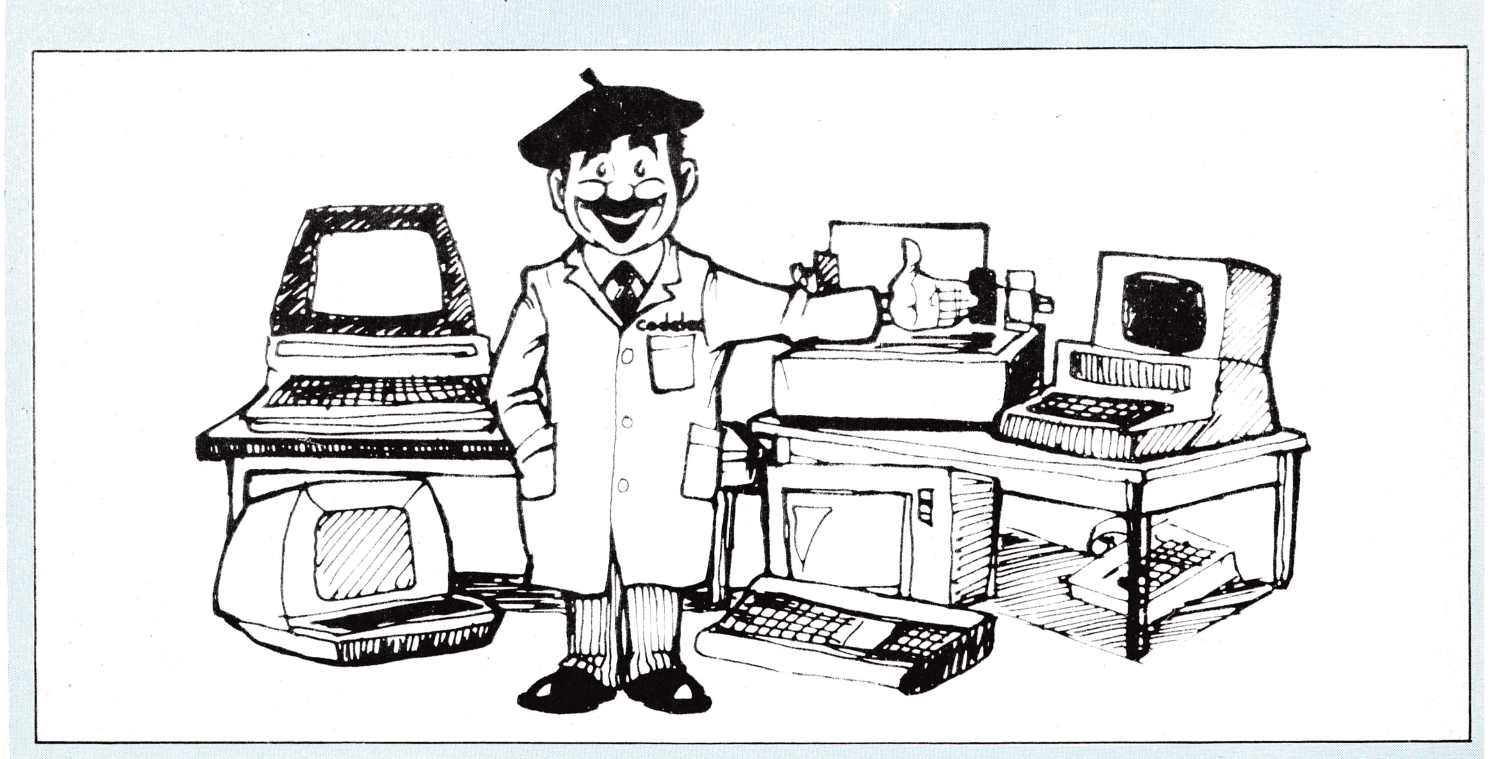
Con lo cual, una vez ejecutada la orden, habrá desaparecido el fichero nombrado EJEMPLO SEI y, en su lugar, tendremos EJEMPLO SIE

El contenido del directorio antes y después de la orden se puede comprobar con un DIR anterior y otro posterior a la misma.

3.° En este nuevo ejemplo asumiremos que el cambio de nombre se ha de producir en un disco instalado en una unidad de disco distinta a la considerada por defecto. La orden, en este caso, puede darse de una de estas tres formas:

```
A> REN B: NOMBRE NUE =
NOMBRE VIE
o
A> REN NOMBRE NUE = B:
NOMBRE VIE
o
A> REN B: NOMBRE NUE = B:
NOMBRE VIE
```

**Comentarios:** Esta orden, como las dos anteriores, sólo actúa sobre la pista catálogo, y los ficheros situados en el área de usuario que a la sazón esté activada.



# COMO CREAR FICHEROS EN DISCO

## Por A. BELLIDO

En otro orden de cosas es interesante conocer un par de peculiaridades.

Por una parte, la velocidad de acceso a una determinada información contenida en disco depende más del tiempo invertido por el cabezal en su movimiento radial que de la velocidad de giro del disco.

Por otra, los sectores ocupados por una información no son consecutivos, pudiéndose dar el caso de que el primero esté situado en la pista 4 y sector 4 y el siguiente en el octavo sector de la misma pista.

Esto es así debido a que la unidad mínima de transferencia, como ya se dijo, entre computador y *drive* es equivalente a la capacidad en bytes de un sector, lo cual implica que antes de producirse la emisión de éstos se requiera un tiempo para llenar el *buffer* o depósito encargado de este trabajo, lo que, a la postre, lleva consigo que el disco se haya desplazado y el siguiente sector disponible no sea el consecutivo.

El *buffer* es una zona de memoria de dimensión predeterminada y ajustada a la función que ha de cumplir.

En nuestro caso, la capacidad del *buffer* corresponde a la de un sector, de forma y manera que la información sea transmitida por grupos de bytes convenidos entre la memoria receptora o emisora y la memoria o receptora la RAM y el disco.

El buffer, en todo caso, actúa como un depósito de bytes que al llenarse procede a actuar del modo que se haya previsto.

La misión fundamental del *buffer* es aumentar la velocidad de transferencia de la información.

Consideremos, en este sentido, que el tiempo consumido en localizar espacio disponible en el disco para el almacenamiento de una información es mucho más elevado que el necesario para transferir esa información de la CPU al disco. Consiguientemente, la operatividad general del sistema gana en eficacia si la transmisión se efectúa en bloques de *bytes* en lugar de *byte* en lugar de *byte* a *byte.*

**RESUMEN DEL PROCESO DE FORMATEADO**

A) Da forma, divide en pistas y sectores, la superficie del disco.
B) Inicializa todos los *bytes* de control de cada sector.
C) Inicializa la pista catálogo con todos los datos necesarios al DOS para manipular el disco.
D) Borra toda la información previa que el disco pudiera contener.
E) El comando FORMAT formatea el disco reservando, según los casos, las pistas 0 y 1 para el DOS.

## FICHEROS. ORGANIZACION

En su momento se dijo que un fichero es un grupo de datos relacionados entre sí, y también se apuntó que el contenido de un fichero depende de la voluntad del usuario.

Ahora nos aproximaremos a la forma en que el programador puede organizar los datos contenidos en un fichero. Y al léxico utilizado.

Para ello, supongamos que se ha decidido controlar por medio de fichas los libros de una biblioteca en función de tres datos: Título, autor y editor.

Las fichas, de cartulina, han sido diseñadas con tres reglones preimpresos, en los cuales se escribirán los datos citados:

Título: ______________________________

Autor: ______________________________

Editor: ______________________________

Para crear el fichero, tendremos que tomar la caja que ha de contener las fichas y colocar en su frontal el título del fichero, para distinguirlo de otros que pueda haber y destinados a otro fin.

Al nuestro lo llamaremos LIBROS.

Con estos precedentes, tomaremos una ficha en blanco y el primer libro cuyos datos se vayan a plasmar en aquella.

Este libro resulta ser "El mus", de Mingote, editado por Prensa Española.

En este punto hagamos una pequeña recapitulación.

Desde que hemos comenzado el ejercicio han aparecido:

—Tres datos "El mus", Mingote y Prensa Española.
— Una ficha: que contendrá los datos anteriores.
— Tres apartados en la ficha: En cada uno de ellos se escribirá el dato correspondiente.
— Un fichero: De nombres LIBROS, en el que se guardarán la ficha presente y el resto de las necesarias para controlar la biblioteca en cuestión.

Expresemos esto mismo en términos informáticos y por idéntico orden:

— Tres datos: "El mus", Mingote y Prensa Española.
— Un registro: Que contendrá los datos anteriores.
— Tres campos: En cada uno de ellos se grabará el dato correspondiente.
— Un fichero: De nombre LIBROS en el que se almacenarán el registro presente y el resto de los necesarios para controlar la biblioteca en cuestión.

Se pueden observar dos cambios. El concepto transmitido por la palabra "ficha" se traspasa al vocablo "registro", y el abarcado por "apartados en la ficha", a "campos".

'Veamos estas nuevas ideas mediante un ejemplo cualquiera en una representación gráfica esquemática.

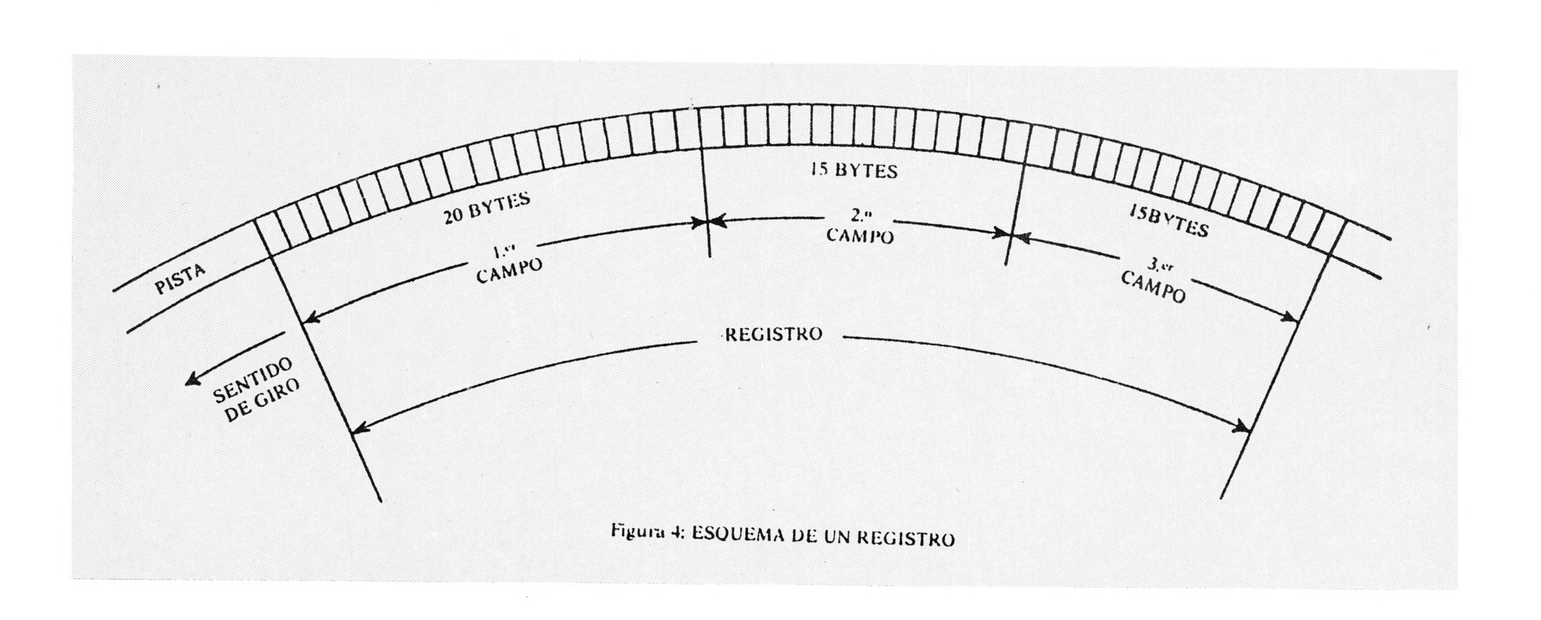



Figura 4: ESQUEMA DE UN REGISTRO

# MATRICES

## REVISION

El término inglés **array** expresa la idea de colocar en un orden preestablecido un grupo de elementos con un fin determinado. Más concretamente, desde un punto de vista matemático implica la organización de números u otros símbolos en filas y columnas, y el programador lo encontrará con frecuencia en ciertos mensajes del ordendor, en artículos y otros publicaciones.

El vocablo español equivalente más apropiado en **matriz** que es una de las acepciones, vale por: "Conjunto de número o símbolo algebraicos colocados en líneas horizontales y verticales y dispuestos en forma de rectángulo".

En términos informáticos una matriz es un conjunto de variables, numéricas o alfanuméricas, con el mismo nombre y subíndice acorde con la posición que ocupa en ella.

Cuando hablamos, p.e., de la matriz ELE o de ELE$ estamos haciendo referencia a una matriz cuyo nombre genérico es ELE y sus elementos en el primer caso, son numéricos y en el segundo alfanuméricos.

Esto mismo representado por medio de la escritura da lugar a ELE ( ) o ELE$ ( ), según el caso. Es decir, el nombre de la matriz seguido de un paréntesis vacío.

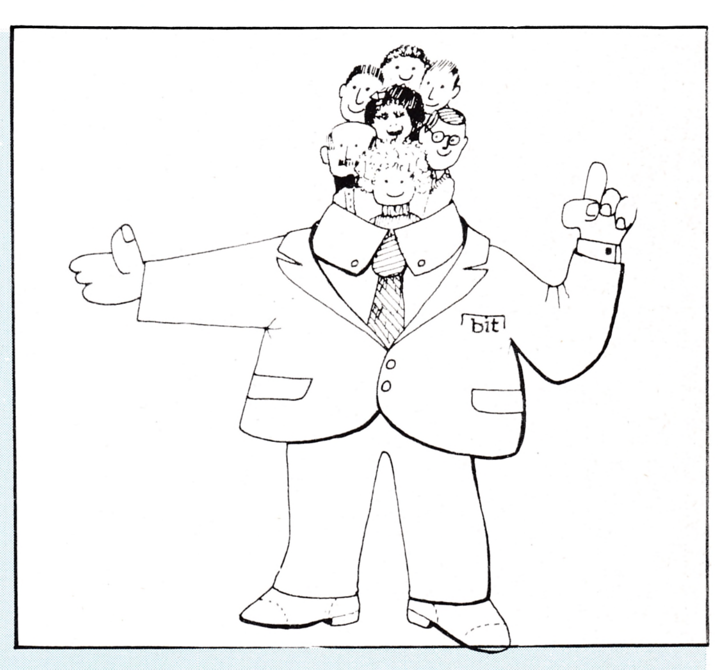
Cada una de las variables que integran la matriz es un elemento de la misma, y como tales variables pueden ser usadas con toda libertad.

Para usar matrices en un programa hay que empezar por dimensionarlas mediante la orden DIM. Por ejemplo:

DIM ELE (12). Con esta instrucción se crean trece variables numéricas de nombre ELE y subíndices correlativos del 0 al 12 de la siguiente forma:

ELE (0), ELE (1), ELE (2), ELE (3), ELE (4), ELE (5), ELE (6), ELE (7), ELE (8), ELE (9), ELE (10), ELE (11), ELE (12).

La instrucción DIM ELE (12) resulta en lo mismo pero con variables alfanuméricas, así: ELE$(0), ELE$(1), ELE$(2), ELE$(3) ELE$ (4), ELE$ (5), ELE$ (6), ELE$ (7), ELE$ (8), ELE$ (9), ELE$ (10), ELE$ (11), ELE$ (12).




Las siguientes notas resumen los puntos más importantes a tener en cuenta al trabajar con matrices, siempre teniendo en cuenta, claro está, las indicaciones del manual del BASIC que se esté utilizando:

**NOTAS:**

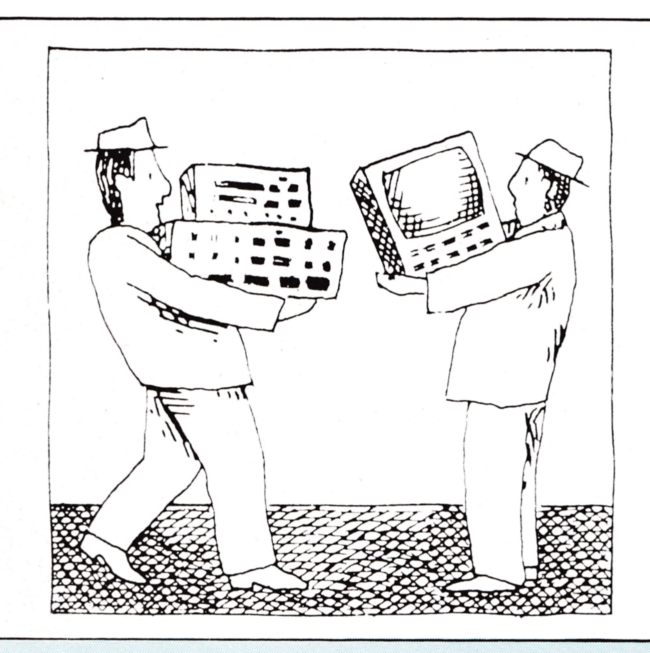
1.° El nombre genérico que designa a una matriz y a los elementos que comprende puede estar compuesto por cualquier combinación de letras y números siempre que el primer caracter sea una letra, y, para las matrices alfanuméricas, el último sea el símbolo "$".

2.° El subíndice más bajo que puede tener el elemento de una matriz es el 0.

3.° Mediante OPTION BASE n, siendo n 0 ó 1, se puede cambiar el subíndice más bajo a 0 ó 1 respectivamente. Esta instrucción, si se usa, ha de ser antes de dimensionar las matrices.
OPTION BASE no suele estar disponible en algunos BASIC.

4.° Hasta el subíndice 10, si se manejan elementos de una matriz que no ha sido dimensionada previamente se sobreentiende una matriz con el nombre dado a ese elemento y de subíndice mayor 10.

5.° Si se hace referencia a una variable cuyo subíndice sea mayor que el más alto indicado al dimensionar la matriz —ó 10 por defecto, según la nota anterior— se obtiene el error **"Subscript out of range"**, o cualquier otro que indique el error cometido con el subíndice.



6.º El mensaje **"Duplicate definition", "Array already dimensioned in"** u otros del mismo porte aparece siempre que se quiere dimensionar una misma matriz por segunda vez.

7.º Mediante la instrucción ERASE nombre matriz1, nombre matriz2,... Se borran las matrices indicadas.

8.º Nada impide utilizar una variable regular (sin subíndice) con el mismo nombre que las variables de una matriz.

9.º Los elementos de una matriz pueden organizarse según una sencuencia lineal de los subíndices —unidimensionales— o distribuyéndolos en varias dimensiones —multidimensionalmente—. Algunos BASIC permiten hasta 255 dimensiones.

10.º Cuando en un programa, o en modo directo, se ejecuta una orden DIM, todas las variables de la matriz en cuestión toman el valor inicial cero en el caso de las numéricas o la cadena vacía en el de las alfanuméricas.

11.º La orden RUN borra las matrices. Por tanto, una vez ejecutado un programa con RUN por primera vez, y si no se quiere perder la información contenida en las matrices, el programa deberá correrse mediante el mandato —en modo directo— GOTO seguido del número de la línea a partir del cual se le quiere expistar y, en todo caso, posterior a las sentencias DIM que se quieren preservar.

## ASIGNACION DE VALORES A LAS VARIABLES CON SUBINDICE

Como quedó dicho en la última nota, tras un orden DIM las variables de la matriz afectada están pendientes de recibir valores. Existen tres métodos fundamentales: Mediante IMPUT, mediante READ/DATA y mediante el volcado de una matriz en otra. Veamos algunos listados que puedan servir de prototipos de los tres casos:

**Mediante INPUT:**

```
10 CLS
20 DIM L$ (14), P (14)
30 P = 0
40 INPUT "Libro"; (L$ (P)
50 INPUT "Precio"; P (P)
60 P = P + 1: IF P  I  = 14 = THEN 40
70 END
```

**Comentarios al listado anterior:**

En la línea 20 se dimensionan dos matrices una alfanumérica —L$( )— y otra numérica —P( )— ambas de 15 elementos: Del L$(0) al L$(14).

En la 30, se inicializa la variable que actuará de controlador gracias al cual los subíndices irán tomando valores. Se ha introducido deliberadamente esta variable —P— con el objeto de dejar claro que pueden subsistir una variable sencilla con el mismo nombre que el correspondiente a las variables con subíndice de una matriz, sin que ello provoque confusión en el BASIC.

En la línea 40, se permite la entrada por teclado a una cadena de caracteres que dará contenido a la variable de nombre genérico L$(P).

En la línea 50 se permite la entrada por teclado a un número que dará valor a la variable numérica P(P).

En la línea 5C aumenta el contador en una unidad con lo cual se actualiza la variable que da valor a los subíndices, comprobándose a continuación si éstos superan el número más alto permitido por la matriz.

**Mediante READ/DATA:**

```
10 CLS
20 DIM L$(14), P(14)
30 P = 0
40 READ L$(P), P(P)
50 P = P + 1: IF P   = 14 THEN 40
60:
1000) DATA EL QUIJOTE, 1200, LA DIVINA COMEDIA,
900, FAUSTO, 950, OTELO, 750, HAMLET, 750, CALIX-
TO Y MELIBEA, 600, LA GALATEA, 500 EL LAZARILLO
DE TORMES, 600, LAS ALEGRES COMADRES DE
WINDSOR, 700, FUENTEOVEJUNA, 800, EL MEJOR
ALCALDE, 800, EL CABALLERO DE OLMEDO, 800, PE-
RIBAÑEZ, 800, EL BUSCON, 800
```

**Comentarios al listado anterior:**

Al igual que en el caso anterior, en la línea 20, se dimensionan las matrices L$( ) y P( ), y en la L40 se inicializa el contador P.

En la 40 se organiza la lectura de los datos contenidos en el DATA de la línea 1100, según lo cual las matrices L$( ) y P( ) irán tomando un título y un precio conforme aumenta el contador.

En la línea 50 se efectúan la actualización del contador y la comprobación de borde oportuna.

**Mediante el volcado de una matriz en otra:**

Para estudiar esta alternativa suponemos que el listado se inicia en la línea 10 del programa del caso anterior, con lo cual la matriz L$( ) ha sido dimensionada y sus elementos valorados.

```
 60:
 70 DIM R$(14)
 80 FOR X =0 to 14
 90 R$(X) = L$(X)
100 NEXT X
```

**Comentarios al listado anterior:**

Tras la ejecución de estas cuatro líneas, los elementos de la matriz R$(  ) han recibido la asignación del contenido de los elementos correspondientes de la matriz L$(  ).

Tras este breve repaso a las matrices, veamos algunas manipulaciones usuales realizadas sobre ellas.

# DESLIZAMIENTOS Y ROTACIONES

ara estudiar los dos conceptos comprendidos en el encabezamiento, vamos a considerar una matriz de nombre A$ y ocho elementos que representaremos así:

| A$(1) | A$(2) | A$(3) | A$(4) | A$(5) | A$(6) | A$(7) | A$(8) |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Según este gráfico, el contenido de los elementos de la matriz A$( ) es la cadena vacía.

Para dar a entender que A$(1) es igual a "A", A$(2) igual a "B", A$(3) igual a "C", etc., se representará de esta forma:

| A$(1) | A$(2) | A$(3) | A$(4) | A$(5) | A$(6) | A$(7) | A$(8) |
|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H |

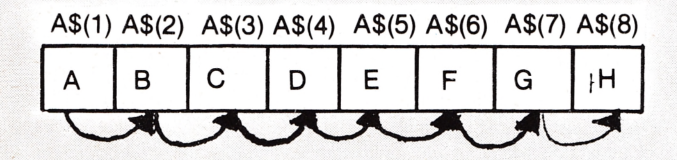
Hechas estas prevenciones, vamos a ver en que consiste un deslizamiento.

## DESLIZAMIENTOS

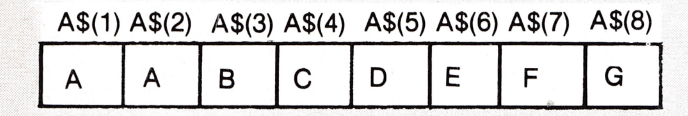
Cuando, por las razones que sean, interesa desplazar el contenido de cada elemento de una matriz a su inmediatamente contiguo, bien a derecha bien a izquierda, se da lugar a un deslizamiento. Esto mismo representado gráficamente, da lugar a la siguiente serie de alternativas:

**Deslizamiento a la derecha**

Antes del deslizamiento:




Después del deslizamiento:




Como se puede observar, en este tipo de deslizamiento a la derecha se pierde el contenido de la variable situada más a la derecha, que equivale a la de subíndice más alto.

El programa que hace posible un deslizamiento a la derecha, en estructura y considerando dimensionada la matriz y valorados sus elementos, es el siguiente:

```
100 FOR X = 3 TO STEP-1
110 A$ (X) = A$(X-1)
120 NEXT X
```

La ejecución de este programa da lugar a la serie de desplazamiento que se indican en la figura, resultando finalmente en el deslizamiento deseado.

| A$(1) | A$(2) | A$(3) | A$(4) | A$(5) | A$(6) | A$(7) | A$(8) | |
|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | Inicial |
| A | B | C | D | E | F | G | G | 1ª vuelta X=8 |
| A | B | C | D | E | F | F | G | 2ª vuelta X=7 |
| A | B | C | D | E | E | F | G | 3ª vuelta X=6 |
| A | B | C | D | D | E | F | G | 4ª vuelta X=5 |
| A | B | C | C | D | E | F | G | 5ª vuelta X=4 |
| A | B | B | C | D | E | F | G | 6ª vuelta X=3 |
| A | A | B | C | D | E | F | G | 7ª vuelta X=2 |

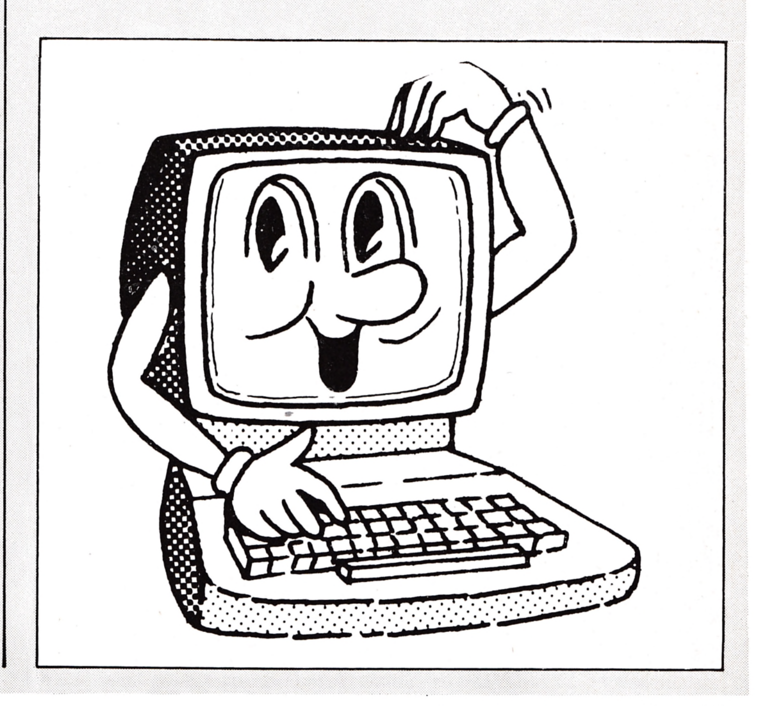
Para conseguir este mismo efecto, pero a la izquierda:

```
100 FOR X = 2 TO 8
110 A$ (X-1) = A$ (X)
120 NEXT
```

La ejecución de este programa da lugar a la serie de desplazamientos que se indican en la figura resultando finalmente en el desplazamiento deseado.

| A$(1) | A$(2) | A$(3) | A$(4) | A$(5) | A$(6) | A$(7) | A$(8) | |
|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | Inicial |
| B | B | C | D | E | F | G | H | 1ª vuelta X=2 |
| B | C | C | D | E | F | G | H | 2ª vuelta X=3 |
| B | C | D | D | E | F | G | H | 3ª vuelta X=4 |
| B | C | D | E | E | F | G | H | 4ª vuelta X=3 |
| B | C | D | E | F | F | G | H | 5ª vuelta X=6 |
| B | C | D | E | F | G | G | H | 6ª vuelta X=7 |
| B | C | D | E | F | G | H | H | 7ª vuelta X=8 |

Para evitar la pérdida del contenido de cualquiera de las variables más extremas, izquierda o derecha, según el tipo de deslizamiento, bastaría asignar previamente ese valor a una variable auxiliar. Esto se puede representar de la siguiente manera:



# EL BASIC DE

Por A. B

## CONFIGURACION DE LOS CAPITULOS

En los capítulos siguientes se estudia el vocabulario BASIC más usual.
Cada capítulo responde a la siguiente estructura:

| PALABRA | SINTAXIS DE LA PALABRA | TRADUCCION |
|---|---|---|

**INTERPRETACION** — **Explica el uso general de la palabra**

**POSIBILIDADES** — **Explica las diferentes posibilidades de utilización**

**FORMA DE TECLEAR LA INSTRUCCION**

**EJEMPLOS** — **Para aclarar el campo de utilización**

**EJERCICIOS** — **Sugiere y/o plantea ejercicios para el aprendizaje.**

Antes de que inicie el estudio del primer capítulo que le ayudará a introducirse en el lenguaje de programación BASIC, permítanos darle...

*Un consejo:* Si no tiene microordenador, procure proveerse de uno. Sin él tendrá dificultades para asimilar cuanto a continuación se expone.

*Otro consejo:* Siempre que piense qué podría suceder si hace tal o cual cosa, ¡NO LO DUDE Y HAGALA!, ésta es la mejor vía para aprender.

*Y un consejo final:* Provéase de un cuaderno y escriba en él todos los ejercicios que realice. Le será muy útil a la hora de confeccionar sus propios programas.

| PRINT | PRINT EXPRESIONES | IMPRIMIR |
|---|---|---|

### INTERPRETACION:

Significa *''imprimir''* y ordena la impresión en pantalla de la expresión o expresiones que figuran, optativamente, en las expresiones.

### POSIBILIDADES:

* Si las expresiones son emitidas, una línea de pantalla salta en blanco.

** Si las expresiones son incluidas y estas expresiones son cadenas de caracteres, debemos escribirlas entrecomilladas.

*** Si las expresiones son incluidas y estas expresiones son matemáticas, aparecerán en pantalla sus valores. Recuerde que las expresiones matemáticas no se escriben entre comillas.

La posición de impresión de cada expresión de expresiones viene determinada por los símbolos que separan una expresión de otra:

— Un punto y coma (;) hace que la siguiente expresión se imprima justo a continuación de la inmediata anterior.

— Una coma (,) obliga a imprimir a partir de la siguiente zona de tabulación o al comienzo de la línea siguiente si ya se ocuparon todas las zonas.

Si las expresiones finalizan con una (,) o un punto y coma (;), la siguiente instrucción PRINT, que pudiera aparecer a lo largo del programa, comenzaría la impresión de sus expresiones conforme al criterio expuesto más arriba.

En caso de no acabar con una coma o un punto y coma, la siguiente instrucción PRINT comenzaría la impresión de sus expresiones en la siguiente línea de pantalla que esté libre.

### FORMA DE TECLEAR LA INSTRUCCION

Teclear PRINT, seguido de ENTER, para dejar en blanco una línea en la pantalla.

Teclear PRINT y la expresión entre comillas ("), seguido de ENTER para que aparezca en pantalla *la expresión alfanumérica.*

Teclear PRINT y la expresión, seguido de ENTER, para que aparezca en pantalla *la expresión numérica.*

### EJEMPLO:

1. Escriba un programa que imprima en pantalla la cadena de caracteres: **"La loba lamía, lentamente, los lobeznos 3 veces".**

*SOLUCION:*

| PROGRAMA | COMENTARIOS |
|---|---|
| 10 PRINT "La Loba" | La forma de teclear esta línea de programa, respetando las normas dictadas por cada MANUAL, sería:<br>• Teclear 10<br>• Teclear PRINT<br>• Teclar "(comillas)<br>• Escribir el texto.<br>• Dar un espacio.<br>• Teclar "(comillas).<br>• Apretar ENTER. |

Una vez seguidas las instrucciones dadas en COMENTARIOS, la línea número 10 de nuestro primer programa estará en la memoria del ordenador. En este caso, tenemos un programa, compuesto por una sola línea, cuya misión es imprimir el texto indicado cada vez que se le ordene al computador.

De una forma similar iríamos tecleando las siguientes líneas del programa:

| PROGRAMA | COMENTARIOS |
|---|---|
| 20 PRINT "lamía, lentamente";<br>30 PRINT "los lobeznos";<br>40 PRINT "3 veces" | • Teclear 20<br>• Teclear PRINT<br>• Teclear "(comillas)<br>• Escribir el texto |

- Dar un espacio
- Teclear " (comillas)
- Teclear; (punto y coma)
- Apretar la tecla ENTER

- Teclear 30
- Teclear PRINT
- Teclear "
- Escribir el texto
- Dar un espacio
- Teclear "
- Teclear ;
- Apretar ENTER

- Teclear 40
- Teclear PRINT
- Teclear
- Escribir el texto
- Teclear "
- Apretar ENTER

Para ordenar al computador que EJECUTE el programa que tiene en memoria, el BASIC dispone del comando RUN.

Así, en el caso que nos ocupa, si tecleamos RUN y, a continuación, ENTER, inmediatamente aparecerá en la pantalla:

*La loba lamía, lentamente, los lobeznos 3 veces.*

Una vez en este punto, es conveniente observar que la definición del ejemplo no fue del todo correcta, ya que decíamos: "Escriba un programa que imprima en...", cuando realmente deberíamos haber dicho: "Escriba un programa que, una vez ejecutado, determine la impresión en pantalla de..."

Esta precisión tiene por objeto fijar las ideas sobre las diferentes expresiones y conceptos que estamos manejando. Pasemos ahora a otros ejemplos.

**EJEMPLO:**

2. Escriba un programa que, una vez ejecutado, imprima en la pantalla la palabra **"Juan"**, que deje la siguiente línea en blanco e imprima en la siguiente **"excelente"**.

*SOLUCION:*

| PROGRAMA | COMENTARIOS |
|---|---|
| **10 PRINT "Juan"**<br>**20 PRINT**<br>**30 PRINT "excelente"** | La única novedad de este programa de tres líneas se presenta en la 20, con el comando PRINT sin ninguna expresión a continuación. Al ser ejecutado el programa, dejará una línea en blanco entre el anterior PRINT y el siguiente:<br><br>Ejecute el programa con RUN y ENTER, como ya sabe. |

**EJEMPLO:**

3. Escriba el programa que, una vez ejecutado, imprima al comienzo de una línea de pantalla "MUY" y, en la siguiente zona de esta línea, "BIEN".

*SOLUCION:*

| PROGRAMA | COMENTARIOS |
|---|---|
| **10 PRINT "MUY", "BIEN"** | La coma introducida entre las dos cadenas entrecomilladas hará que la segunda cadena comience su impresión al principio de la siguiente zona de la misma línea.<br>Ejecute el programa. |

**EJEMPLO:**

4. ¿Cómo se producirá la impresión en pantalla de las cadenas de caracteres que figuran en el siguiente programa una vez ejecutado?

*SOLUCION:*

| PROGRAMA | COMENTARIOS |
|---|---|
| **10 PRINT "JUNTO"**<br>**20 PRINT "A TI"**<br>**JUNTO A TI** | El punto y coma con que acabamos la primera instrucción PRINT obliga a la expresión PRINT siguiente a imprimirse justo a continuación de la anterior.<br>Observe que se ha dejado deliberadamente un espacio al final de la primera cadena (línea 10). |

**EJEMPLO:**

5. Escriba un programa que, una vez ejecutado, imprima en pantalla el resultado de sumar 2 y 2.

*SOLUCION:*

| PROGRAMA | COMENTARIOS |
|---|---|
| **10 PRINT 2 + 2** | Como observará, la expresión que sigue al comando PRINT va sin comillas, ya que es una expresión matemática. Las comillas sólo se usan para indicar al ordenador que deseamos que una expresión sea tratada |

como una cadena de caracteres.
En el caso que nos ocupa, ordenamos al computador que imprima el resultado de la operación indicada.

**EJEMPLO:**

6. Escriba un programa que, una vez ejecutado, imprima en pantalla: 2 + 2 *son 4.*

*SOLUCIONES:*

| PROGRAMA | COMENTARIOS |
|---|---|
| **10 PRINT "2 + 2 son"; 2 + 2**<br>**o también**<br>**10 PRINT "2 + 2 son ",**<br>**20 PRINT 2 + 2** | La primera expresión, por ser entrecomillada, será tratada como una cadena de caracteres.<br>Recuerde que, entre el último caracter de la cadena y las comillas, debe dejar un espacio. Así evitará que al ejecutar el programa aparezca en pantalla 2 = 2 *son 4.* ¡Cuide la correción en la escritura! El punto y coma que separa las dos expresiones PRINT obliga al ordenador a que imprima el resultado de la operación matemática justo a continuación del último caracter de la cadena anterior. |

Para dejar la pantalla limpia de los caracteres impresos con anterioridad, se dispone del comando CLS.

Ambos serán estudiados posteriormente, pero hasta ese momento, usted puede **limpiar** la pantalla tecleando **CLS** y **ENTER.**

Realizar dibujos esquemáticos:

1.
```
10 PRINT "XXXXXX"
20 PRINT "X    X"
30 PRINT "X    X"
40 PRINT "X    X"
50 PRINT "X    X"
60 PRINT "XXXXXX"
```
2. Trace las diagonales del cuadrado anterior.
3. Borre las diagonales del cuadrado anterior y uno de los lados. Transfórmelo en un rectángulo de doble base que altura.
4. Dibuje un triángulo rectángulo de catetos iguales.
5. Dibuje libremente.

Es conveniente diseñar los dibujos en un papel milimetrado antes de imprimirlos en pantalla: evitará muchos errores y pérdidas de tiempo... ¡y de paciencia!

*SOLUCIONES:*

2.
```
1 REM Ficha "PRINT" Ej.: 3.2"
10 PRINT "XXXXXX"
20 PRINT "XX   X"
30 PRINT "X X  X"
40 PRINT "X  X X"
50 PRINT "X   XX"
60 PRINT "XXXXXX"
```

3.
```
1 REM Ficha "PRINT" Ej. 3.3.
10 PRINT "XXXXXXXXXXXX"
20 PRINT "X          X"
30 PRINT "X          X"
40 PRINT "X          X"
50 PRINT "X          X"
60 PRINT "XXXXXXXXXXXX"
```

4.
```
1 REM Ficha "PRINT" Ej. 3.4
10 PRINT "X"
20 PRINT "XX"
30 PRINT "X X"
40 PRINT "X  X"
50 PRINT "X   X"
60 PRINT "X    X"
70 PRINT "X     X"
80 PRINT "X      X"
90 PRINT "X       X"
100 PRINT "XXXXXXXXXX"
```

# FICHAS DEL
## GUIA DE PAL

### *IF comparación THEN sentencia*

Donde el resultado de **comparación** sólo puede ser **cierto o falso.**

Obliga a pasar la secuencia de lectura a **sentencia** si el resultado de la comparación es **cierto,** en otro caso continúa en la siguiente línea de programa.

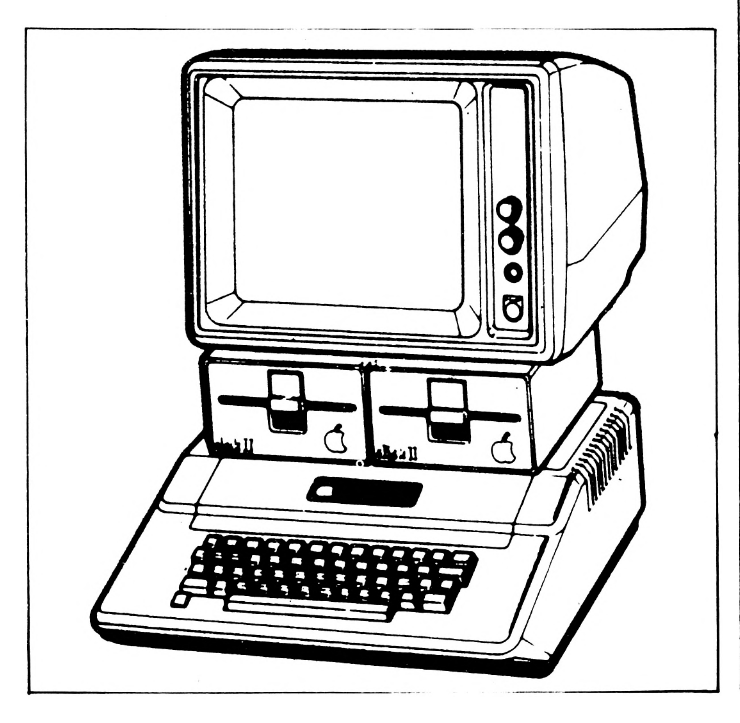
```
10 INPUT A
20 IF A = 3 THEN PRINT "TRES"
30 PRINT "NO ES TRES".
```

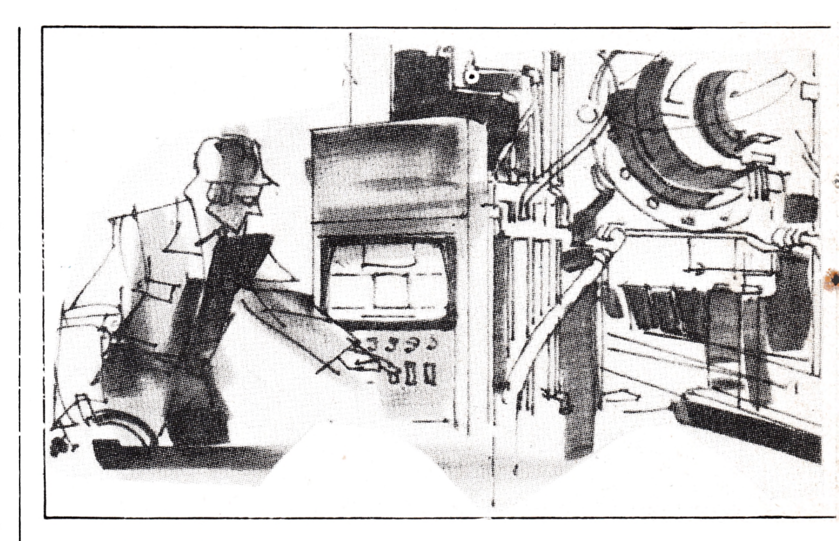
### *INKEY$*

Capta el caracter de la tecla apretada en el momento de ser procesada esta instrucción, si no hubiera ninguna, se considera la cadena vacía.

```
10 PRINT "Apretar una tecla".
20 IF INKEY$ = " "THEN GOTO 20
50 PRINT "Tecla apretada".
```

En la línea 20 se produce un bucle sobre sí misma, ya que mientras no se apriete una tecla, el computador capta la cadena vacía, y consiguientemente, pasa la lectura a la línea 20.





### *INPUT "texto"; ¡Variable!*

Provoca una parada en el programa para, de esta forma, dar un contenido a la variable.

Si el "texto" ha sido escrito, éste aparecerá en la pantalla durante la interrupción y hasta que el dato solicitado haya sido tecleado y se apriete la tecla ENTER.

Si el "texto" no existe, sólo el **prompt** (con la expresión **prompt** se viene a indicar que el ordenador está preparado para hacer su trabajo y, para ello, hace aparecer en pantalla un símbolo, al que por extensión se llama así) típico del ordenador aparecerá en pantalla, indicando que está a la espera de información. En este caso, el dato que se introduzca será asignado a la variable.

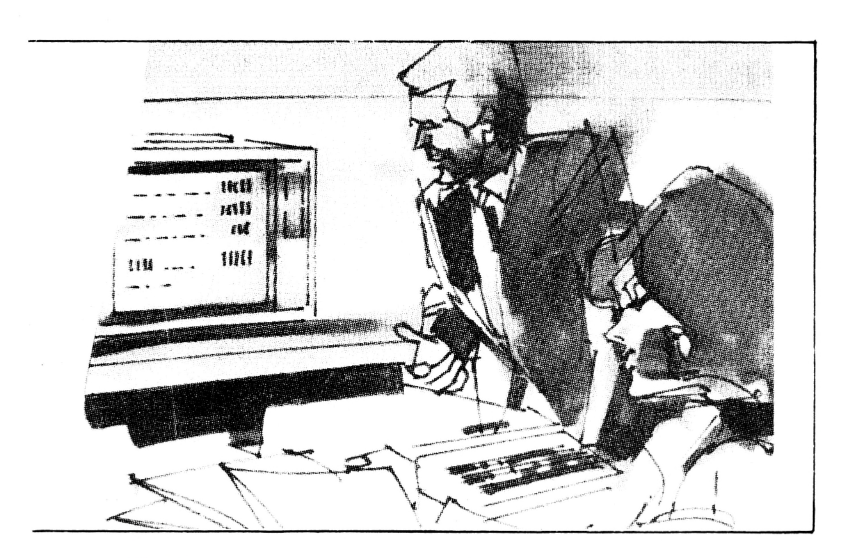
Esta variable puede ser numérica o de caracteres y, consiguientemente, el dato introducido tiene que estar en consonancia con el tipo de variable. Es decir, si la variable se ha considerado como numérica, no se aceptará una cadena de caracteres como entrada del INPUT.

Escriba un programa que, una vez ejecutado, nos pida el número de "pasos" consumidos en una conferencia telefónica y, a continuación, el valor de cada "paso" para obtener finalmente el importe de la conferencia.

```
10 INPUT "¿Número de pasos?"; N
20 INPUT "Valor de unpaso?"; P
30 PRINT "El importe de su conferencia es";
40 PRINT N*P.
```

Con las dos primeras líneas provocaremos, al ejecutar el programa, dos interrupciones para asignar valores a las variables N y P.

# ...ABRAS BASIC



### INSTR (A$,B$)

Localiza y muestra la posición donde comienza la primera coincidencia de la cadena de caracteres B$ dentro de la cadena A$.

```
10 $ = "ABCD"
20 B$ = "C"
30 PRINT INSTR (A$, B$)
```

El resultado es 3.

### INT ¡expresión!

Se calcula el menor de los números enteros del valor que da —o es— **expresión.**

```
10 PRINT INT 1,2
20 PRINT INT —1,2
```

### LEN ¡expresión!

Calcula el número de caracteres que componen la cadena definida en **"expresión".**

```
PRINT LEN "PLUS".
```

### LET ¡variable = expresión

Asigna a una variable el valor de una expresión.

Recordemos previamente que una **variable** contiene un **valor** que puede cambiar a lo largo de un algoritmo, pero que permanece fijo mientras no haya una instrucción que así lo determine.

Existen dos clases de variables: **numéricas** y **de caracteres.**

**Las variable numéricas** se representan por cualquier conjunto de caracteres —a veces, por una sola letra seguida de un número— con la condición de que el primero sea una letra. Ejemplo: LET ABc = 45.

**Las variables de caracteres** se representan por una sola letra seguida del símbolo $. De esta forma, el computador sabe que el contenido de la variable sería un texto.

### LEFT$ ¡("expresión", expresión)!

Extrae una subcadena de **"expresión"** formada por tantos caracteres com indique la **"expresión"** numérica y contados desde la izquierda.

### LIST ¡expresión1 · expresión2!

Proporciona un listado en pantalla del programa que actualmente esté en memoria, y, a partir de la línea de programa que indique **expresiones.** Si las **expresiones** no se escriben entonces se obtiene el listado completo.

### LOAD ¡"expresión"!

Carga desde el disco a la memoria el programa cuyo nombre viene dado por **"expresión".**

### LOCATE ¡expresión1, expresión2!

Posiciona el cursor en la columna dada por **expresión1** y fila dada por **expresión2.**

### LOG ¡(expresión)!

Calcula el logaritmo natural —en base e— del valor que da —o es— **expresión.**

```
PRINT LN 1,1.
```

### LOG10 ¡expresión!

Calcula el logaritmo —en base 10— del valor que da —o es— **expresión.**

```
PRINT LN 1,1.
```

# MASTE

**Este mes presentamos una versión sencilla del conocido juego "Master Mind". Como seguramente conocerá se trata de adivinar una clave, —en este caso numérica—, de forma que además de acertar los números que la componen, estos deben estar en la debida posición.**

## REGLAS DEL JUEGO

Si un determinado número existe en la clave pero no se encuentra en la posición exacta, el ordenador lo indicara con el símbolo 0 , si el número existe y además se encuentra en su sitio lo indicara con , en el otro caso posible, número no existente en la clave el ordenador no mostrará ningún signo, es decir, sólo se señalan los aciertos.

Los aciertos son totalizados en cada jugada y señalados a la derecha del número introducido, pero en ningún caso la posición de estos símbolos coincide con posiciones reales de los números, es decir indica únicamente cuántos números hay acertados y cuántos son plenos, pero no cuales lo son.

La clave está compuesta por cuatro cifras numéricas, cada una de ellas puede ser un número comprendido entre 1 y 9, estos números pueden repetirse e incluso en un caso límite (poco probable) pueden ser todos iguales.

Para la obtención de los cuatro números que componen la clave se utiliza un procedimiento aleatorio. El ordenador elabora la clave y nosotros debemos adivinarla.

## COMENTARIOS

Dado que suponemos que el lector conoce las instrucciones más elementales del BASIC, como PRINT, REM, etc., únicamente comentaremos aquellas líneas o grupos de líneas más importantes.

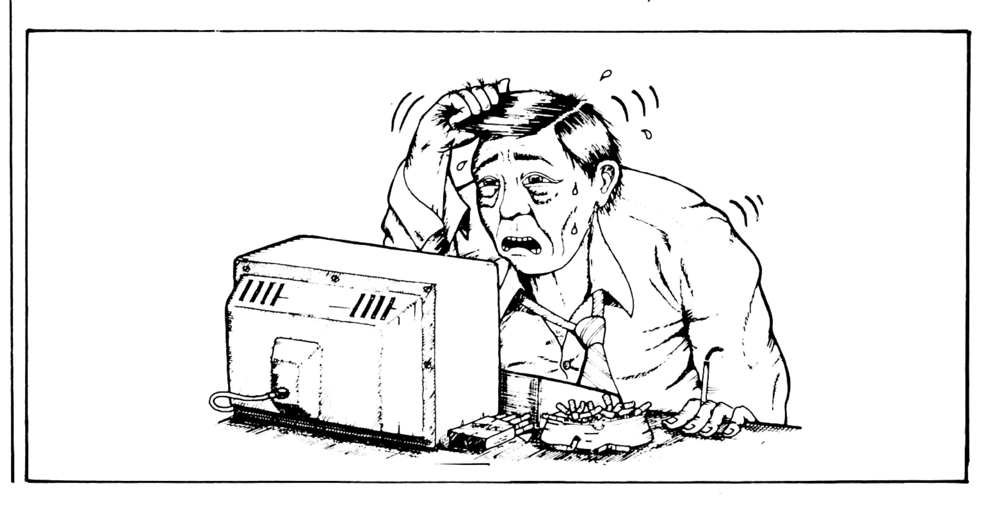
30 Establece una ventana en la parte inferior de la pantalla para la emisión de mensajes por la misma. Ocupa los 40 caracteres de la línea 25 (última de la pantalla).

60-70 CHR$ (164) Es el símbolo de acierto pleno, y CHR$ (230) el de número sólo. Ambos son caracteres standard del Amstrad, incluidos en el manual.

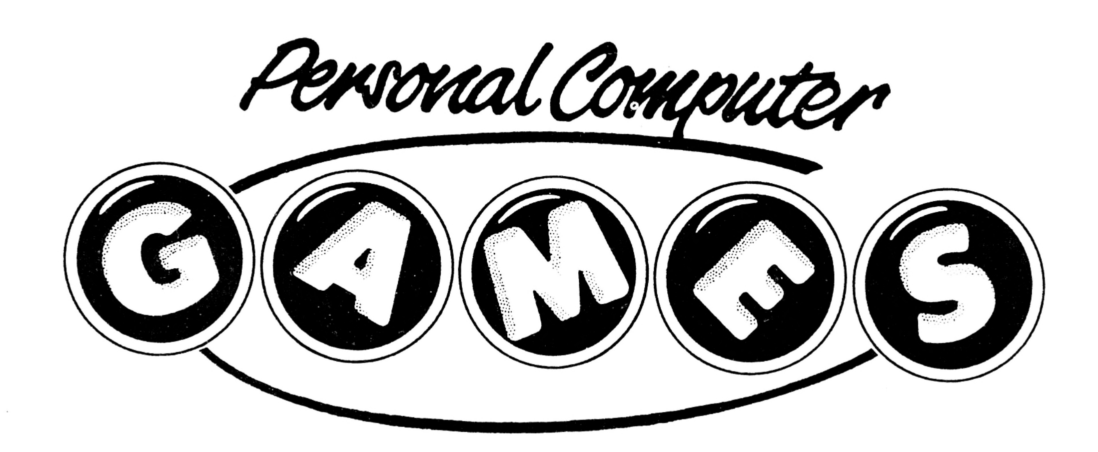
80 Mensaje en ventana, precedido por un pitido provocado por CHR$ (7). La llamada a la rutina &BB18 provoca una interrupción del programa hasta que se presione una tecla.

105 CHR$(22) Controla la impresión transparente (sobreimpresión), con CHR$(1) se activa y con CHR$(0) se anule.

120 Mejora la aleatoriedad del proceso de generación de la clave, al hacer depender ésta del la var. TIME (tiempo que lleva encendido el ordenador).



# R MIND




130 Introduce un número aleatorio generado por RND, comprendido entre 1 y 9 en la matriz (ver. indexada) CLAVE, que no es previamente dimensionada, ya que sólo tiene 4 elementos ( 10).

170 Bucle para la entrada de los cuatro números de cada jugada.

180-190 Entrada de los números de la jugada, en forma de cadena (textos y no números propiamente), utilizando la función INKEY$ en vez de INPUT.

200-210 Si el caracter introducido es válido, se almacena en la matriz NUM, que tiene cuatro elementos como la CLAVE. La función VAL$ permite el paso de los números introducidos como textos a números propiamente dichos.

220-225 Confirmación de si los números introducidos en la jugada son los deseados (evita los errores al teclear).

255-270 Si los números no eran válidos, se hace retroceder el cursor hasta la posición inicial, para introducir nuevamente la jugada, tal como la indica el mensaje de la 270.

280 Si los números introducidos son válidos, a poner a cero unos contadores de aciertos P y Q.

300 Si un número coincide con uno de la clave y en posición, se incrementan los contadores y a la variable T$ se le agregan los símbolos correspondientes.

320 Si T$ contiene 4 caracteres se translada el control a la línea 500, a fin de comprobar si ha acertado la clave o no.

330-400 Se comprueba la existencia de números iguales a los de la clave, aunque no estén en su sitio.

420 La var. CONTADOR controla el número de jugadas, establecido en un máximo de 10. Si es menor podemos seguir jugando por la transferencia a la línea 160.

430 En caso contrario, —se acabó el tiempo—, y el ordenador muestra la clave no acertada. El CHR$ (24) invierte los colores del vídeo.

500-520 Controla la existencia o no de acierto pleno (los cuatro números y en su sitio), en caso contrario devuelve el control a la 410 para proseguir el juego si no se han agotado las jugadas disponibles.

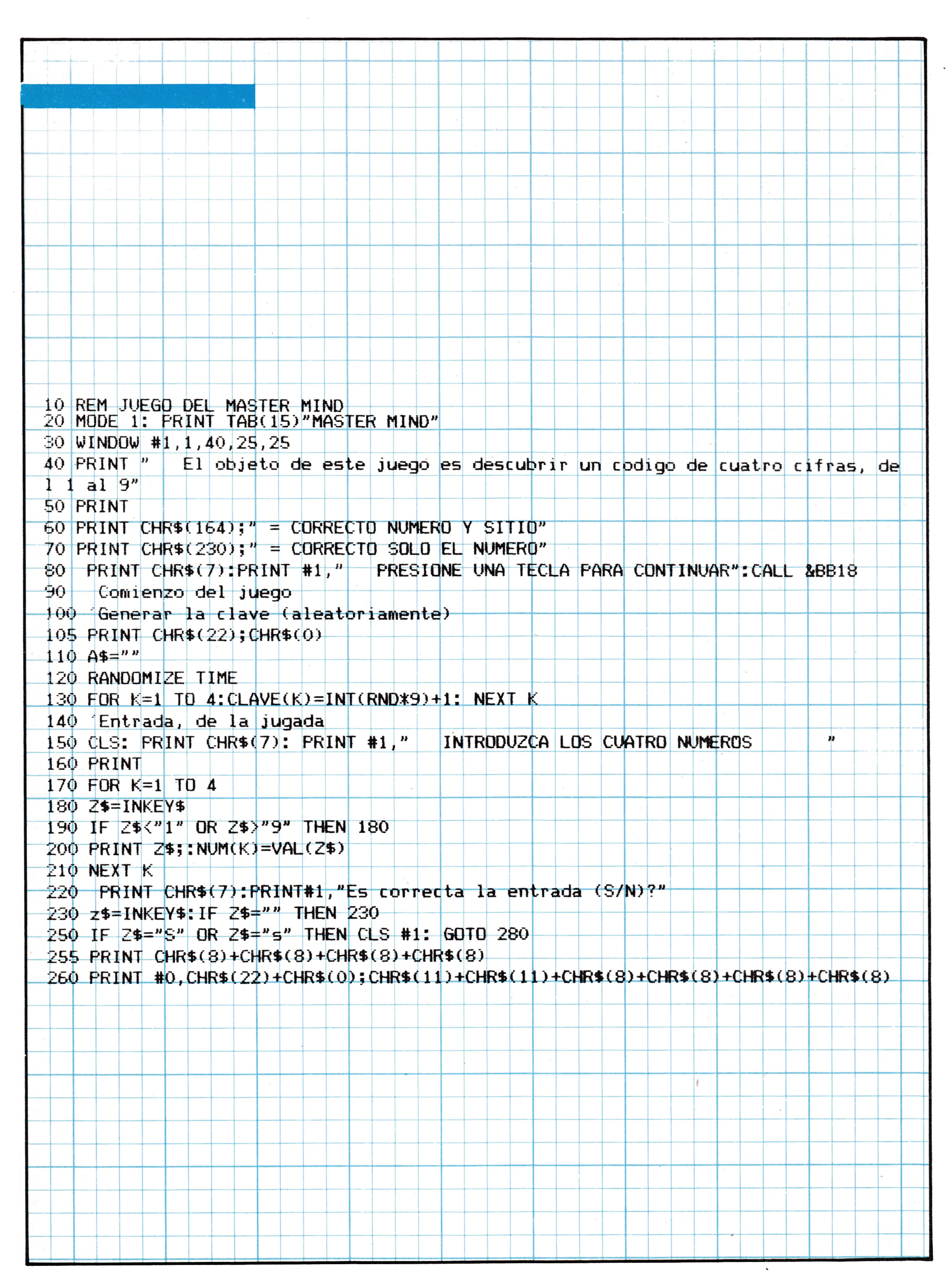
## MEJORAS

Además de las derivadas de una mejor presentación, puede incrementarse el nivel de dificultad añadiendo un dígito más, es decir que los números vayan del 0 al 9. Modifique para ello las líneas.

40 del 0 al 9
130 RND * 10
190 IF Z$ "0"

Los caracteres utilizados para mostrar los aciertos de las jugadas puede cambiarlos por otros más a su gusto, elíjalos en el manual, tome nota de su código y cambie estos valores en las líneas 60, 70, 300 y 500.

Si dispone de monitor en color puede mejorar mucho la presentación, añadiendo una buena combinación de colores.



```
10 REM JUEGO DEL MASTER MIND
20 MODE 1: PRINT TAB(15)"MASTER MIND"
30 WINDOW #1,1,40,25,25
40 PRINT "   El objeto de este juego es descubrir un codigo de cuatro cifras, de
l 1 al 9"
50 PRINT
60 PRINT CHR$(164);" = CORRECTO NUMERO Y SITIO"
70 PRINT CHR$(230);" = CORRECTO SOLO EL NUMERO"
80  PRINT CHR$(7):PRINT #1,"   PRESIONE UNA TECLA PARA CONTINUAR":CALL &BB18
90   Comienzo del juego
100 'Generar la clave (aleatoriamente)
105 PRINT CHR$(22);CHR$(0)
110 A$=""
120 RANDOMIZE TIME
130 FOR K=1 TO 4:CLAVE(K)=INT(RND*9)+1: NEXT K
140 'Entrada, de la jugada
150 CLS: PRINT CHR$(7): PRINT #1,"   INTRODUZCA LOS CUATRO NUMEROS       "
160 PRINT
170 FOR K=1 TO 4
180 Z$=INKEY$
190 IF Z$<"1" OR Z$>"9" THEN 180
200 PRINT Z$;:NUM(K)=VAL(Z$)
210 NEXT K
220  PRINT CHR$(7):PRINT#1,"Es correcta la entrada (S/N)?"
230 z$=INKEY$:IF Z$="" THEN 230
250 IF Z$="S" OR Z$="s" THEN CLS #1: GOTO 280
255 PRINT CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)
260 PRINT #0,CHR$(22)+CHR$(0);CHR$(11)+CHR$(11)+CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)
```

```
270 CLS #1:PRINT CHR$(7):PRINT #1,"     INTRODUZCA DENUEVO LOS NUMEROS      ":GOTO
 170
280 T$="":FOR K=1 TO 4:P(K)=0:Q(K)=0: NEXT K
290 FOR K=1 TO 4
300 IF NUM(K)=CLAVE(K) THEN T$=T$+CHR$(164):P(K)=1:Q(K)=1
310 NEXT K
320 IF LEN (T$)=4 THEN  PRINT CHR$(11);CHR$(22);CHR$(1);"       ";T$: GOTO 500
330 FOR I=1 TO 4
340 IF P(I)<>0 THEN 400
350 FOR J=1 TO 4
360 IF Q(J)<>0 THEN 380
370 IF T<>J AND NUM(J)=CLAVE(I) THEN T$=T$+CHR$(230):Q(J)=J:J=4
380 NEXT J
400 NEXT I
410 PRINT CHR$(11);CHR$(22);CHR$(1);"        ";T$
420 IF CONTADOR<10 THEN CONTADOR=CONTADOR+1:PRINT #1,"     INTRODUZCA LOS NUEVOS
NUMEROS      ": GOTO 160
430 PRINT #1,CHR$(24);" LO SIENTO LA CLAVE ERA ";CLAVE(1);CLAVE(2);CLAVE(3);CLAV
E(4);CHR$(24)
440 END
500 FOR K=1 TO 40:IF MID$(T$,K,1)=CHR$(164) THEN  AC=AC+1: NEXT K
510 IF AC=4 THEN  PRINT #1,CHR$(24);" ACERTO A LOS ";CONTADOR;" INTENTOS ";CHR$(
24):END
520 GOTO 410
```

# LA PANTALLA DE TEXTO

Por JUAN MARTINEZ PINTOR

***En el capítulo I se vieron los conceptos de palabra y lista. Se dijo entonces que una palabra es una serie de caracteres consecutivos ninguno de los cuales puede ser un separador y que una lista es una serie de objetos-logo con indicación expresa de donde comienza y donde termina siendo, a su vez, un objeto-logo con indicación expresa una palabra o una lista. Resulta pues que una lista elemental sería la formada exclusivamente por palabras. Esta lista de nivel 1 podría ser uno de los objetos (de los elementos), constituyentes de otra lista (nivel 2), en la que otros objetos podrían ser palabras o listas. Se indicó también entonces que el orden en que estén los elementos de la lista es fundamental.***

**EJEMPLO IV.1.**

[CASA LAPIZ MESA] es una lista cuyos elementos son CASA el primero, LAPIZ el segundo y MESA el tercero. Por ello, [FOLIO [CASA LAPIZ)] es una lista cuyo primer elemento es la palabra FOLIO y cuyo segundo elemento es la lista anterior. Lógicamente la lista anterior y la lista [(FOLIO) (CASA LAPIZ MESA)] son distintas, dado que en el primer elemento de esta última es también una lista pero formada por un sólo elemento, que es una palabra. Todos los elementos de la última lista son, a su vez, listas.

Como Logo es un lenguaje extraordinario capacitado para manejar símbolos, es preciso en muchas ocasiones distinguir entre, por ejemplo, una palabra tomada como tal, y la misma palabra tomada como el objeto representado por ela. Existen órdenes que aplicadas a la palabra como tal producen un resul-

tado muy distinto del obtenido aplicándola al objeto representado. Tendremos ocasión de ver muchos ejemplos de ello.

En consecuencia existe un modo para indicar a Dr. Logo que una palabra ha de ser tomada ella misma como sujeto de la orden. Este modo consiste sencillamente en anteponer el carácter **comillas** (") a la palabra. Esto mismo puede pensarse teniendo en cuenta el concepto de operación estudiado en el punto II.1: Cuando una palabra va precedida por el caracter ", se ejecuta una operación sobre ella cuyo resultado es precisamente el objeto-logo que consiste en esa palabra precisamente. Del mismo modo podemos pensar en los caracteres de delimitación de listas ([ y ]).

### EJEMPLO IV.2

Por ello si se escribe directamente en el ordenador, por ejemplo, **fd 50** la orden será ejecutada de inmediato porque se toma la palabra fd como lo que representa (un primitivo); pero si se escribe **[fd 50]** entonces se escribe esta lista en pantalla sin que ninguna acción se lleve a cabo (en según que versiones de Logo podría ni tan siquiera escribirse en pantalla la lista). Parecido resultado se obtiene escribiendo **"fd 50"** porque fd es tomado ahora como la palabra y no como el primitivo que representa.

De todo lo anterior podemos sacar la conclusión de que Logo dispone de muchos instrumentos para manejar texto considerado como tal. En efecto, ello es así y poco a poco iremos teniendo ocasión de comprobarlo.

## IV. 2. IMPRESION DE TEXTOS EN PANTALLA

En el punto III.3 se ha visto que la pantalla puede estar en uno de los estados siguientes: sólo texto, sólo gráficos o una mezcla de texto en todos los modos posibles de la pantalla incluso entre los gráficos. Existen instrucciones que permiten escribir textos gráficos). Vamos a ver a continuación las órdenes que permiten escribir en las pantallas de texto y mixta.

La instrucción
**pr**
requiere un objeto-logo como dato sobre el que actuar y, cuando se ejecuta, el objeto es escrito en pantalla.

### EJEMPLO IV.3.

Si se ejecuta la orden
pr "Hola
en pantalla se escribe la palabra "Hola". Del mismo modo la instrucción,
pr [Hola buenos días]
escribe todos los elementos de la lista anterior en la pantalla. Obsérvese que el resultado de ejecutar esta orden no incluye la escritura de los corchetes exteriores de la lista. Por el contrario, la instrucción,
pr [Hola buenos días [tenga usted]]
no escribe los corchetes exteriores de la lista principal, pero si los de la lista interior; el resultado sería:
Hola buenos días [tenga usted].

En realidad el primitivo pr admite como dato una lista. De este modo la lista más exterior no ha de delimitarse entre corchetes, y el resultado será que se escribirán en pantalla todos los objetos de la lista pero después de cada uno de los cuales el cursor pasará a la línea inmediatamente inferior (se produce un retorno de carro).

### EJEMPLO IV.4.

Si ejecutamos,
pr "Hola "buenos "días
el resultado será
Hola
buenos
días
pero si ejecutamos
pr "Hola "buenos "días [tenga usted]
el resultado será
Hola
buenos
días
[tenga usted]

Observemos en el ejemplo anterior que cada palabra ha se hacerse preceder del caracter "para que no se interprete, cuando no se incluyen en una lista delimitada con corchetes. Para que no se ejecute un retorno de carro tras la impresión de cada uno de los elemento, basta encerrar toda la orden entre paréntesis (incluso la palabra pr). De este modo, sólo después de escrita toda la lista que se pase a pr como dato, se ejecutará el retorno de carro.

### EJEMPLO IV.5

(pr "Hola "buenos "días [tenga usted])
escribe en pantalla
Hola buenos días tenga usted
y el cursor pasa entonces a la línea inferior.

Todo lo dicho para la orden pr es válido para el primitivo
**show**
con la única diferencia de que este último muestra en pantalla los paréntesis exteriores de la lista. Ya se verá que ésto puede ser de gran utilidad para el estudio de procedimientos, cuando se tratan como listas.

Por lo que a presentación de texto respecta, en la pantalla de texto, sólo resta la instrucción.

**type**

cuya única diferencia con pr es que type no ejecuta un retorno de carro después de ejecutada. Ello significa que dos órdenes type consecutivas presentarían sus resultados en la misma línea (si ello es posible), uno a continuación del otro. Hemos, no obstante, de tomar nota de que el comportamiento anteriormente descrito de type se refiere a su ejecución en un procedimiento. En nivel superior el comportamiento es exactamente igual que el de pr.

Aunque el desarrollo completo de las operaciones numéricas será desarrollado por completo en el capítulo XIV, es ésta una buena ocasión para presentarlas y así poder realizar cálculos con Dr. Logo. En efecto, se dispone de las operaciones aritméticas elementales, designadas con los caracteres:

+ para la suma,
— para la resta,
* para el producto y
/ para la división.

**EJEMPLO IV.6.**

```
pr 2 + 5
7
pr 10 — 4
6
pr 4 * 3
12
pr 120 / 60
2
```

El orden de prioridad de las operaciones así como el uso de paréntesis puede hacerse en Dr. Logo del mismo modo que se hace en otros lenguajes. De momento no insistimos más en ello.

## IV.3. MODOS DE LA PANTALLA MIXTA

Como se vio en el punto III.3, y se ha recordado más arriba, existen tres posibles estados de la pantalla. La pantalla mixta es capaz de albergar gráficos y reserva las líneas más bajas de la pantalla para texto. A la pantalla mixta se accede con la orden ss como ya sabemos e inicialmente el número de líneas que se reservan para texto es 5. Lógicamente, si la tortuga entra en la zona de texto desaparece de la vista, pero continúa obedeciendo las instrucciones que reciba.

El número de líneas de texto que deben reservarse en la pantalla mixta puede modificarse con el primitivo.

**setsplit N**

donde N representa el número de líneas que se reservarán; debe estar comprendido entre 1 y 25. Si se escribe un número fuera del rango anterior se recibe el mensaje de error:

**setsplit doesn't like 0 as input**

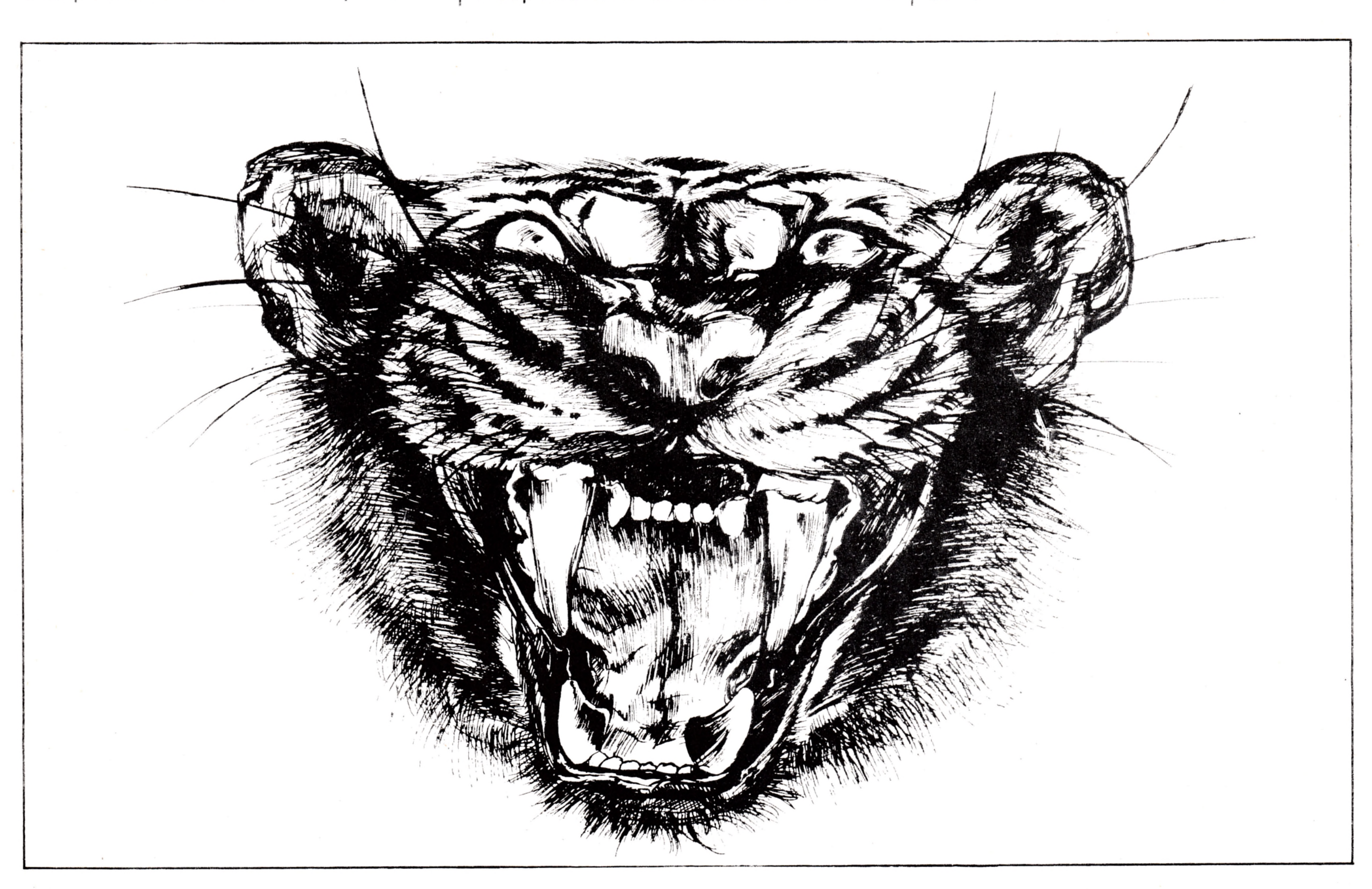
y el número manejado con esta instrucción no será alterado.

Observemos la gran utilidad que puede representar esta orden para escribir programas en los que el número de líneas de texto sea importante. Podemos citar como ejemplo de ello el programa DISPAROS correspondiente al número 2 de la revista. Estudiando su ejecución, y su listado, vemos que se ha usado una sola línea final para pedir los datos del disparo, reservando el resto de la pantalla para el movimiento de la tortuga.

Por otra parte, resulta que aunque existe una orden expresa para borrar los gráficos (cs y clean como se vio en el punto III.7), si con la orden setplit se invaden parte o todos los gráficos, esta parte será borrada; es decir, que no sólo desaparecerán de la vista al ser ocultada por las líneas de texto; si posteriormente se regresa al número de líneas de texto previas a la ejecución de la orden setsplit, los gráficos correspondientes a la zona invadida ya no estarán allí.

Para finalizar el capítulo indicamos que la instrucción siguiente borra los textos de la página de texto o pantalla mixta situando el cursor en la esquina superior izquierda de la zona correspondiente. De modo que en pantalla mixta, el cursor será situado en la primera de sus líneas. La orden a que nos referimos es:

**ct**



# PROGRAMA COMENTADO EN LOGO

# MASTER MIND

## Por JUAN MARTINEZ PINTOR

l siguiente programa desarrolla una parte del juego: el Dr. Logo "piensa" un número que el jugador debe averiguar. El juego se completará en otro número desarrollando la parte en que es el jugador quien piensa el número y Dr. Logo quien intenta averiguarlo.

No se han utilizado gráficos tortuga para representar ninguna situación, pero se sugiere al lector que piense algún tipo de gráfico tortuga que vaya progresando a la vez que se averiguan números.

Se observará al jugar que si el número buscado contiene un determinado dígito varias veces (como el 8 en el número 88238) entonces será considerado herido (o muerto) tantas veces como aparece. Así si se escribe el número 678000 se obtendrá un resultado de 3 heridos.

En el siguiente listado se han separado con una línea los procedimientos de que consta el programa para mayor claridad (no forman parte del listado). Igualmente debe tenerse en cuenta que se han incluido comentarios a las instrucciones del programa encerrándolos entre las secuencias de caracteres (* y *) como se hace en Pascal. *Ninguno de estos caracteres forma parte del programa.* Deben pues ignorarse todas las secuencias de caracteres encerradas entre (* y *) al escribir el programa en el ordenador.

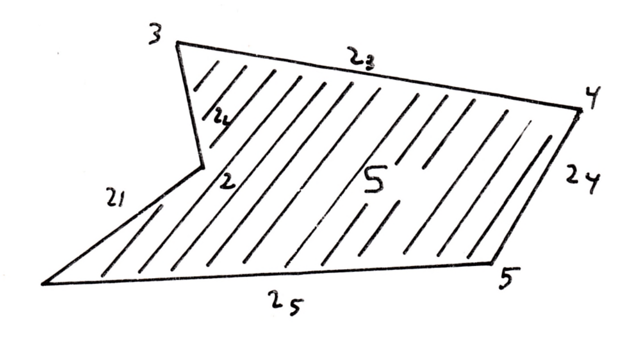
```
to master.mind      (* Definicion del procedimiento principal *)
presentacion        (* Procedimiento invocado *)
ct                  (* Primitivo que borra los textos *)
jugar               (* Llamada a procedimiento *)
end                 (* Concluye el modo definicion *)
------------
to  presentacion         (* Esta instruccion situa a Dr.  Logo en  el
                            modo  definicion de procedimientos.  Va a
                            definirse el procedimiento llamado
                            "presentacion. *)
ts ct                     (* ts selecciona la pantalla de texto y ct borra
                             los textos *)
pr [MASTER MIND]          (* Escribe en pantalla *)
repeat  4[pr []]          (* Repite 4 veces la escritura de una linea  en
                             blanco *)
pr [En cada jugada hay que escribir]
pr [un numero de 5 cifras, todas las]
pr [cuales pueden ser 0]
pr []
pr [En cada jugada se informa del]
pr [numero de muertos y heridos]
repeat 5 [pr[]]
type  [tecla  para seguir]          (* Escribe sin seguir de  retorno  de
                                       carro *)
make  "seguir  rc                   (* Espera hasta que  se  pulse  un
                                       caracter en el teclado y despues
                                       lo asigna a la variable "seguir *)
end
------------
to jugar
coge.num          (* Llama al procedimiento "coge.num *)
label "a          (* Define este punto como la etiqueta "a, para poder
                     enviar aqui el control *)
pedir.num         (* Llamada a procedimiento *)
comprobar (list :x1 :x2 :x3 :x4 :x5) (list :y1 :y2 :y3 :y4 :y5) 1 1
(* Llamada al procedimiento llamado  comprobar. Este  procedimiento,
como puede verse mas abajo,  tiene 4 inputs: dos listas y dos numeros,
por ello su llamada indica las listas y los numeros *)
```

# CALCULO DE AREAS Y PERIMETROS POR COORDENADAS

*Víctor J. Campo López*

**Bajo este título presentamos este mes un programa que permite calcular el área y perímetro de una figura plana irregular, así como la longitud de cada uno de sus lados, a partir de las coordenadas cartesianas de sus vértices.**

En una poligonal como se indica en la figura, cada uno de los vértices viene definido por una pareja de coordenadas, referidas a unos ejes cartesianos X e Y. Si numeramos estos vértices comenzando por el más próximo al origen y en sentido de las agujas del reloj, obtendremos todos los datos que son necesarios para el programa, a partir de ellos, se obtendrá como se ha indicado el área encenada por la poligonal, el perímetro de la misma y la longitud de cada uno de sus lados.




nº puntos = 5 + 1 = 6

## OBSERVACIONES:

1) Para el programa el número de vértices es el total numerados más uno, ya que el primero se cuenta como origen y como fin, es decir dos veces.

2) Aunque se ha indicado que la numeración debe hacerse en sentido de las agujas del reloj, puede realizarse en sentido contrario, siempre que los puntos sean correlativos, la única diferencia reside en que el área vendrá expresada con un número positivo o negativo, pero idéntico si se toma el valor absoluto, si se desea puede modificarse la línea 470 añadiendo ABS (área/2).

3) A partir de la línea 500 se realiza un dibujo de la poligonal con indicación de sus vértices, a fin de visualizar la forma gráfica de dicha poligonal. Si no se desea utilizar esta parte pueden suprimirse las líneas 480 a 640, final del programa.

FORMULACION EMPLEADA:

La superficie viene dada por:

$$\$ = \sum_{i=2}^{i=n+1} (X_{i-1} - X_i) * (Y_{i-1} + Y_i))/2$$

La longitud de los lados es:

$$L_i = \sqrt{(X_{i-1} - X_i)^2 + (Y_{i-1} + Y_i)^2}$$

Y el perímetro será: $P = \sum L_i$

```
if  :muerto<5 [(pr "muertos :muerto "heridos :herido) pr [] go  "a][pr
"correcto]]
(*  Condicional.  Si la variable "muerto" tiene un valor menor que  5,
entonces  se escribe el numero de muertos y heridos,  se  escribe  una
linea  en  blanco y se transfiere el control al punto marcado  por  la
etiqueta "a (label "a).  En otro caso se escribe "correcto" y el juego
termina *)
end
------------
to coge.num
make "x1 random 10 make "x2 random 10 make "x3 random 10
(*  Asignaciones  de valores a las  variables  "x1..."x5.  Se  escogen
numeros al azar entre 0 y 9 *)
make "x4 random 10 make x5 random 10
end
------------
to pedir.num
pr []
(pr "escribir "el "numero)
make "y1 rc type :y1
(*  Se  pide al teclado el contenido de la variable  "y1.  Despues  se
imprime en pantalla sin que se siga con retorno de carro *)
make "y2 rc type :y2
make "y3 rc type :y3
make "y4 rc type :y4
make "y5 rc type :y5
pr []
make  "muerto  0
(*  Antes de comenzar las comprobaciones,  se inicializa el numero  de
muertos y heridos a 0 *)
make "herido 0
end
------------
to comprobar :lista1 :lista2 :n :m
(* El procedimiento "comprobar lleva como datos de entrada las  listas
que  ha  de  comprobar  y los numeros de los elementos  de  ellas  que
comprueba si son iguales *)
if item :n :lista1 = item :m :lista2 [make "test "i][make "test "d]
(*  Si el elemento numero :n de la lista llamada "lista1 es  igual  al
elemento  numero :m de la lista "lista2 entonces se asigna el valor "i
(de iguales) a la variable "test; de otro modo se asigna "d *)
if  :test="i  [if  :n=:m  [make  "muerto  :muerto+1] [make  "herido
:herido+1]]
(*  Si  la  variable "test contiene al valor "i entonces  si  en  esta
activacion  de "comprobar  es :n igual a :m se aumenta  el  numero  de
muertos de la jugada; si :n y :m son distintos se aumenta el numero de
heridos *)
if :m<5 [make "m :m+1][make "m 1 make "n :n+1]
(* Si el valor de la variable "m es menor que 5 en esta activacion del
procedimiento "comprobar, entonces se aumenta "m en 1; de otro modo se
asigna a "m el valor 1 y se aumenta "n en 1 *)
if or :n<5 :n=5 [comprobar :lista1 :lista2 :n :m]
(* Siempre que :n sea menor o igual que 5, el procedimiento "comprobar
se  llama a si mismo pero con los nuevos valores de "n y  "m.  Resulta
pues que este procedimiento es recursivo. *)
end
------------
```

```
10 REM CALCULO DE AREAS Y PERIMETROS
              POR COORDENADAS
20 ' Victor J. Campo Lopez  May-86
30 MODE 1
40 LOCATE 7,5: PRINT CHR$(24);"CALCULO D
E AREAS Y PERIMETROS"
50 LOCATE 12,7: PRINT "POR COORDENADAS";
CHR$(24)
60 LOCATE 1,12
70 PRINT "PARA INTRODUCIR EL NUMERO DE P
UNTOS DEBE  CONSIDERARSE EL ORIGEN DOS V
ECES, COMO  PUNTO DE PARTIDA Y COMO LLEG
ADA."
80 PRINT : PRINT
90 DEF FN A(P)=(X(P-1)-X(P))*(Y(P-1)+Y(P
))
100 DEF FN D(P)=SQR((X(P-1)-X(P))^2+(Y(P
-1)-Y(P))^2)
110 INPUT "NUMERO DE PUNTOS";N
120 DIM X(N),Y(N),DIS(N)
130 CLS: PRINT "INTRODUCIR COORDENADAS D
E LOS PUNTOS"
140 PRINT "-----------------------------
----------"
150  FOR P=1 TO N-1
160 PRINT "PUNTO";P;
170 INPUT "X,Y";X(P),Y(P)
180 NEXT P
190 PRINT "PUNTO";N;"cierre";x(1);y(1)
200 x(n)=x(1):y(n)=y(1)
205 PRINT : PRINT "PRESIONE UNA TECLA PA
RA VERIFICAR DATOS"
206 CALL &BB18
210 MODE 2: PRINT TAB(18)"RELACION DE CO
ORDENADAS DE LOS PUNTOS"
220 WINDOW #1,1,40,24,25
230 PRINT "  PUNTO"TAB(15)"X";TAB(28)"Y"
;TAB(42)"PUNTO";TAB(54)"X";TAB(67)"Y"
240 PRINT "=============================
========================================
=========="
250 FOR P=1 TO N:PRINT "    ";P,X(P),Y(P)
,: NEXT P
260 PRINT #0
270 PRINT #1,"D A T O S      C O R R E C
 T O S   (S/N)"
```

```
280 Z$=INKEY$:IF Z$="" THEN 280
290 Z$=UPPER$(Z$):IF Z$="S" THEN 360
300 CLS #1: INPUT #1,"CUANTOS PUNTOS ERR
ONEOS ";D
310 FOR P=1 TO D
320 CLS #0
330 INPUT "INTRODUCIR: PUNTO, X, Y ";PE,
X(PE),Y(PE)
340 NEXT P
350 GOTO 210
360 FOR P=2 TO N
370 AREA=AREA+FN A(P)
380 DIS (P)=FN D(P)
390 DISTANCIA=DISTANCIA+ FN D(P)
400 NEXT P
410 MODE 1
420 PRINT "PUNTO  DISTANCIA"
430 PRINT "================"
440 PRINT 1;TAB(9)" ----"
450 FOR P=2 TO N: PRINT P;:PRINT TAB(6)U
SING "####.###";DIS(P): NEXT P
460 PRINT
470 PRINT "PERIMETRO=";DISTANCIA,"AREA="
;AREA/2
480 LOCATE 20,2: PRINT "PRESIONE TECLA"
490 IF INKEY$="" THEN 490
500 REM     D I B U J O
510 PRINT CHR$(22);CHR$(1)
520 MODE 2
530 XMAX=X(1):YMAX=Y(1)
540 TAG
550 FOR P=2 TO N
560 IF X(P)>XMAX THEN XMAX=X(P)
570 IF Y(P)>YMAX THEN YMAX=Y(P)
580 NEXT P
590 KX=620/XMAX:KY=380/YMAX
600 FOR P=2 TO N:PLOT X(P-1)*KX,Y(P-1)*K
Y:DRAW X(P)*KX,Y(P)*KY
610 MOVER 5,12:IF P=N THEN  PRINT "1"; E
LSE PRINT P;
615 NEXT
620 TAGOFF
630 PRINT CHR$(22);CHR$(0)
640 IF INKEY$="" THEN 640
```

# NUESTRO PRÓXIMO NÚMERO




N.º 5

EL AMSTRAD Y EL CMP

Ficheros en el AMSTRAD

Basic del AMSTRAD

Fichas del AMSTRAD

Programando en BASIC

Logo del AMSTRAD

Programa en LOGO

El Programa técnico del mes

